AÑO II N.º 20

MICROHOBBY AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO II N.º 20

160 Ptas. (incluido I.V.A.)

LOS SECRETOS DEL TECLADO: COMO MODIFICARLO DE PRINCIPIO A FIN

CP/M: HISTORIA DE UN STANDARD

Organiza tus datos en programas basic: las matrices

ATERRIZA COMO PUEDAS CON AMSTER

SOFTWARE

Highway encounter: alcanzar la zona cero

HOBBY PRESS, S.A.

microbyte PRESENTA... AMSTRAD

NUEVOS PROGRAMAS EN CASSETTE Y DISCO

ARGO NAVIS

El comandante de nave AMSTRAD. Se encuentra atrapado en las profundidades de una central nuclear y debe salir con vida. Excelentes gráficas y sonido. P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.

ROCK RAID

Debes pilotar con acierto la nave que a lo largo de su viaje galáctico sufrirá encuentros con meteoritos, residuos planetarios, etc. Gran movilidad y excelentes efectos. P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.

WIZARD'S LAIR

Te encuentras atrapado en las profundidades de una caverna, llena de obstáculos, adversidades, etc. ¿Serás capaz de salir con vida? P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.

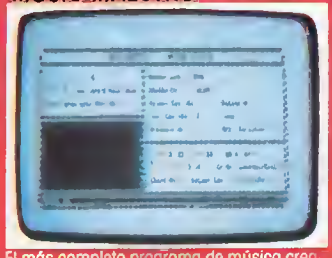
MACADAM FLIPPER

Atractivo programa que nos traslada al manejo de la máquina-flipper del mejor casino de Las Vegas. Posibilidad de creación de tablero, puntuaciones, etc. P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.

JUMP JET

Te encuentras a los mandos de la nave "Airgraff". En una perfecta maniobra debes despegar del portaviones. (Excelente versión simulador vuelo-combate). P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.

MUSIC MAESTRO



El más completo programa de música creado para el AMSTRAD. Permite crear sonidos, melodías y convertir tu ordenador en la mejor "caja de música". P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.

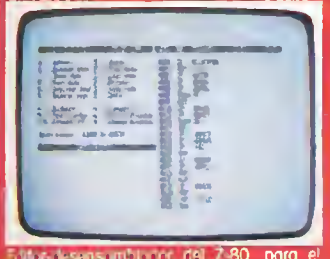
PAZAZZ

Programa que permite de una manera sencilla la creación de pantallas con gráficos, dotarles de movimiento, acompañados de música. P.V.P.: DISCO 2.900 pts.

SYCLONE 2

Programa de utilidad que permite realizar copias de seguridad (backups) a distintas velocidades (bauds). P.V.P.: CASSETTE 1.800 pts. DISCO 2.500 pts.

ZEDIS II



Editor-desensamblador del Z-80, para el programador más avanzado. P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.

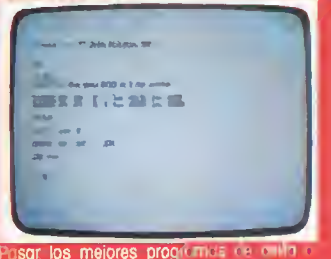
SYSTEM X

Ampliación del lenguaje Basic. Conjunto de 30 nuevas instrucciones (fill, circle, protec, para ayudar en la programación. P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.

ODDJOB

La mejor utilidad para el mejor conocimiento del disco.
(Copias de disco, Disk map, Disk track, sector, etc.). P.V.P.: DISCO 2.600 pts.

TRANSMAT

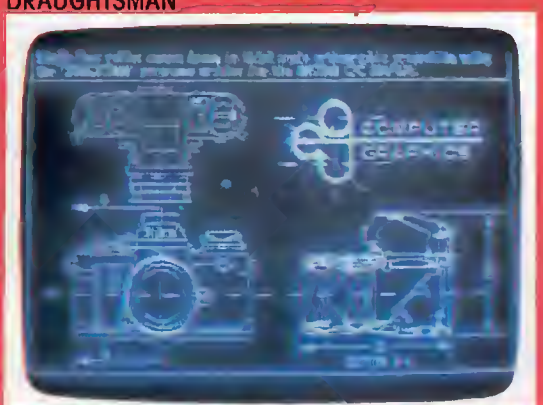


Pasar los mejores programas de cinta a disco ya no es problema. Con Transmat todo el proceso será fácil y sencillo. P.V.P.: DISCO 2.600 pts.

OTROS PROGRAMAS EN STOCK

MINI OFFICE	P.V.P. CASS. 3.200 pts. P.V.P. DIS. 3.900 pts.
WORLD CUP FOOTBALL	P.V.P. CASS. 1.800 pts.
BATLE FOR MIDWAY	P.V.P. CASS. 1.800 pts.
FIGHTER PILOT	P.V.P. CASS. 2.200 pts.
SURVIVOR	P.V.P. CASS. 1.800 pts.
MOON BUGGY	P.V.P. CASS. 1.800 pts.
TECHNICIAN TED	P.V.P. CASS. 1.800 pts.
FRUITY FRANK	P.V.P. CASS. 1.800 pts.
DATABASE	P.V.P. CASS. 2.100 pts.
LOGO TURTLE GRAPHICS	P.V.P. CASS. 2.400 pts.
TASCOPY Y TASPRINT	P.V.P. CASS. 2.600 pts.
FONT EDITOR	P.V.P. CASS. 1.900 pts.

DRAUGHTSMAN



Sofisticado programa de dibujo que permite tratar la pantalla del AMSTRAD como un sencillo tablero de dibujo, sus resultados son espectaculares. P.V.P.: CASSETTE 4.500 pts. DISCO 5.200 pts.

ENVIENOS A **MICROBYTE** S.A.
P.º Castellana, 179, 1.º - 28046 Madrid

Nombre
Apellidos
Dirección
Población
D.P. Teléfono

ENVIOS GRATIS

JUEGO	C	D	Precio	TOTAL

PRECIO TOTAL PESETAS

Incluyo talón nominativo ☐
Contra-Reembolso ☐

Pedidos por teléfono **91 - 442 54 33 / 44**

Director Editorial
José I. Gómez-Centurión
Director Ejecutiva
Víctor Prieto
Subdirector
José María Díaz
Redactora Jefe
Marta García
Diseño
José Flores
Colaboradores
Francisco Portala
Pedro Sudán
Miguel Sepúlveda
Francisca Martín
Jesús Alonso
Pedro S. Pérez
Amalia Gómez
Juan J. Martínez
Secretaria Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Javier Martínez
Portada
J. Igual
Ilustradores
Javier Igual, J. Pons, F. L. Frontán, J. Septien, Peja, J. J. Mora, Luigi Pérez

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Publicidad
Concha Gutiérrez
Publicidad Barcelona
José Galán Cortes
Tel: (93) 303 10 22/313 71 62

Secretaría de Dirección
Marisa Cagarra

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

Redacción, Administración y Publicidad
La Granja, s/n
Polígono Industrial de Alcobendas
Tel.: 654 32 11
Telex: 49 480 HOPR

Dto. Circulación
Carlos Peropadre

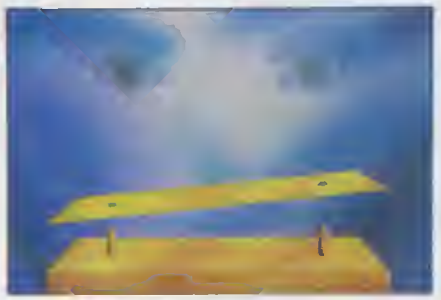
Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
ROTEDIC, S. A. Crta. de Irún. Km. 12,450 (MADRID)
Fotocomposición
Navacamp, S.A.
Nicolás Morales, 38-40
Fotomecánica
GROF
Ezequiel Solana, 16
Depósito Legal:
M-28468-1985

MICROHOBBY AMSTRAD

sumario

Año II • Número 20 • 14 al 20 de Enero de 1986
160 ptos. (Incluido el IVA)



MICROHOBBY

AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

LE OFRECE AHORA SUS PROGRAMAS YA GRABADOS PARA QUE VD. NO TENGA QUE TECLEARLOS. TOTALMENTE DESPROTEGIDOS CON EL OBJETO DE FACILITAR SU COPIA EN DISCO.

Todos los programadores y aficionados a la microinformática, sabemos lo tedioso y propenso a errores que resulta el teclear un listado de un programa. Para facilitar tu labor al máximo y evitar que malgastes largas horas sobre el teclado de tu ordenador tratando de descifrar incomprensiblemente mensajes de error.

AMSTRAD SEMANAL te ofrece cada mes los programas publicados en los cuatro números correspondientes, en una cinta de cassette desprotegida, que te permitirá copiar los programas en disco y tener acceso a los listados para su estudio y posterior edición de rutinas.

Programas incluidos en la cinta número 1

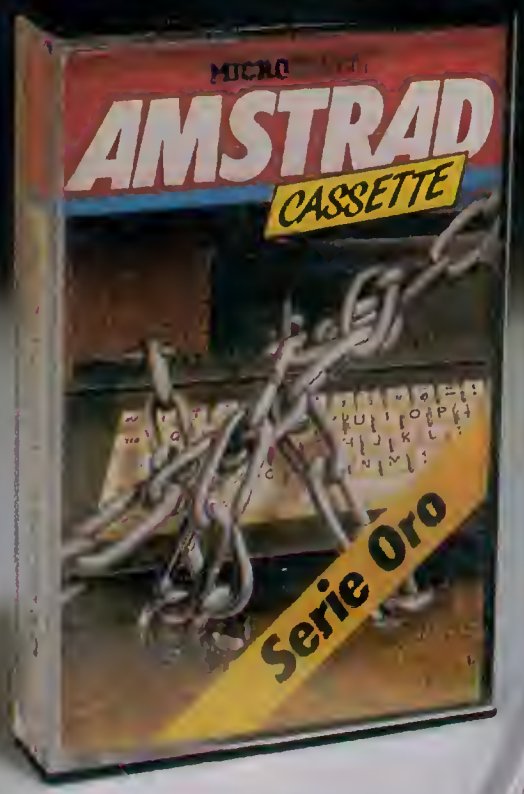
Título	Rev. n	Título	Rev. n.
EASYDRAW	1	MAD ADDER	3
EGGBLITZ	2	HEXER	3
CODIGO SECRETO	2	CHARGEN	4
VENTANAS	2	PROGRAMACION	4
BIORRITMOS	3		

Programas incluidos en la cinta número 2

Título	Rev. n.	Título	Rev. n.
GRAFICOS	8	INCOGNITON	7
MUSICA	6	MONITOR	5
TRON	6	ANALISIS	5 8
ENSAMBLADOR	8	CEDRIC	8
HEXERL	8	ANIMACION1	7
TOOLKIT	8	ANIMACION2	8
PRIMEROS PASOS	7	SMALEY	7

Programas incluidos en la cinta número 3

Título	Rev. n.	Título	Rev. n.
ANALISIS	9-12	SPRITE12	11
FRUTAS	9	AMSCARD	11
MENUDISC	9	OTELO	12
RSx101	10	EVENT121	12
RSx102	10	EVENT122	12
CUATRORAYA	10	FRUTTIES	12
SPRITE11	11		



Por sólo 675 pts.
(incluidos gastos de envío)

COMPATIBLES CON LOS MODELOS CPC-464, CPC-664 y CPC-6128

Recíbelos en tu casa cómodamente enviándonos con la menor demora posible, el cupón que se encuentra en la última página de la revista

Primera plana

Pasado y futuro, juntos hoy

No cabe duda de que la informática está tomando un puesto de primera línea en las actividades más insólitas, facilitándolas y haciéndolas más versátiles cada día que pasa.

Y si no, que se lo digan al grupo de arqueólogos de la Universidad de Cambridge, que están trabajando en unas excavaciones que se remontan a las postrimerías de la Edad del Bronce en Gubbio, Italia central.

Esta gente ha instalado, en Cambridge y Gubbio, ordenadores personales M24 de Olivetti, que encima están conectados a los portátiles M10, también de Olivetti.

La red de ordenadores hará posible un tratamiento instantáneo de la ingente cantidad de información que debe procesarse en una excavación arqueológica, contando con la inestimable ayuda de una conexión directa con las bases de datos de la universidad inglesa. Bien por Cambridge y por Olivetti.

La batalla de los superordenadores

Como ya hemos comentado antes en esta sección de Primera plana, el ordenador más potente del mundo se llama Cray-1, y está fabricado por Cray Research, cuyo presidente es Seymour Cray **(¡sorpresa!).**

A IBM esto no le parece muy bien, y ha anunciado la fabricación de un supercomputador capaz de procesar vectores. No está muy claro lo que esto quiere decir, pero debe ser importante porque Seymour Cray se ha apresurado a declarar en París que ellos harán lo mismo, antes y mejor, y que potenciarán al máximo las ventas de su sistema.

Cray Research posee varios acuerdos de investigación conjunta con IBM, pero no parece estar involucrada en la idea del procesador vectorial de la multinacional americana. El campo de los superordenadores debe ser uno de los pocos de la informática en los que IBM no dicta su ley. Por ahora.

STARMOUSE, NUEVO RATON PARA AMSTRAD

Todos los amantes de la informática, conocen las grandes posibilidades y la facilidad de utilización, que ofrecen los ordenadores de tipo Macintosh.

El ratón, es el elemento físico que posibilita la comunicación entre el usuario y el programa.

Desplazándolo sobre la mesa de trabajo, se sitúa el cursor sobre el gráfico de la opción deseada y con una simple pulsación, la máquina, ejecuta nuestras órdenes.

En el afán por facilitar el uso de lo ordenadores, todas las marcas introducen sistemas operativos que funcionan a base de ratón y cómo no, en el caso de **Amstrad**, también podemos disponer de un ratón para dibujar en pantalla.

El aparato se denomina Starmouse, fabricado por Puricorp.

Con unos cuantos minutos de manejo del mismo, se obtienen unos dibujos increíbles.

Los iconos *(menús gráficos de opciones)*, que maneja el Starmouse, permiten operar con el disco, con cinta, hacer una copia impresa de la pantalla, borrar ésta, etc., con la facilidad de un toque de botón y sin escribir ni una palabra.

El icono de opciones de dibujo, es bastante completo, permitiendo borrar, aerógrafo, brocha con distintos gruesos, desde un simple punto, a trazos más gruesos.

Este ratón, es compatible con todos los modelos de **Amstrad** y constituye una herramienta de gran utilidad y un verdadero pasatiempo, para todo aquél que desee dibujar con su ordenador.

HS-S1, INTERFACE EN SERIE DE HONEYSOFT

La pequeña firma Británica, especialista en periféricos para el **Amstrad,** nos sorprende con un producto de increíble valor para todos aquellos que deseen sacar más partido a su máquina.

Este inferface en serie, permite conectar al ordenador impresoras en serie, o la más espectacular de las posibilidades; la utilización de modems, permitiendo la comunicación de el ordenador, con bases de datos y demás servicios informáticos vía telefónica.

El interface es compacto y está dotado de un conector de 25 vías, también se incluye una conexión para futuras expansiones. El control del mismo se puede realizar bajo CPM, o con el uso de comandos Basic, contenidos en ROM, disco o cinta, dependiendo del soporte que decidamos elegir.

El uso del mismo se facilita mediante la aparición en pantalla, de los diversos menús de opciones. Para fijar el formato de pantalla, disponemos de dos modos:

Modo 0, con letra comprimida pudiendo elegir entre ocho colores.

Modo 1, con los caracteres de tamaño normal.

Otras facilidades, que permiten aprovechar al máximo el periférico, son volcado en pantalla de texto o gráficos, grabar páginas de información, reloj digital.

El sistema está preparado para ser totalmente compatile con el servicio de Micronet.

Este software adicional, también está disponible en ROM, disco o cinta, esperándose que versiones mejoradas del mismo, volcados de pantalla en color, y correo electrónico.

El periférico es totalmente compatible con el propio interface de **Amstrad.**

Y AHORA LAS MATRICES

En ocasiones en programación es muy necesario agrupar datos de una misma especie organizados bajo una misma estructura, de forma que podamos acceder fácilmente a uno de ellos en concreto sin variar para nada los demás ni tener que examinar todos ellos. Para eso están las «matrices». Veamos su definición y manejo.

Javier Igual

Echemos un vistazo al programa 1. **¿Le reconoce?** Se trata del programa 5 del número anterior. Sí, el que nos iba a convertir en millonarios. **¿Tuvo suerte al utilizarlo?** Si no ha sido así, no se desanime, esta semana vamos a intentar mejorarlo.

En él vamos paso a paso calculando cada uno de los números de nuestra apuesta y si es válido lo metemos en una de las 6 variables reservadas para cada uno de ellos **«número»** a **«número 6».**

Después, para escribirlo, necesitamos tantas instrucciones PRINT como variables haya —en este caso 6—. Como sólo son 6 variables no se nota mucho el derroche de instrucciones Basic que estamos haciendo, pero, **¿puede imaginar qué pasaría si en lugar de 6 fueran 6.000?** Sería una tarea de chinos confeccionar un programa que diera un valor a esas 6.000 variables diferentes, y además seguramente, ¡se nos acabarían los nombres con que bautizarlas!

Organizar los datos mediante matrices

Vamos a evitar esto construyendo una estructura —*array o matriz*— que sea capaz de contener los valores de estos 6 números del Loto o 6.000 si fuera preciso. Es necesario añadir al principio del programa una línea semejante a:

DIM número (6)

Al verla, nuestro ordenador reservará rápidamente en su memoria una serie de posiciones *(6 en nuestro caso)* para almacenar un conjunto de elementos que hemos agrupado bajo el nombre **«número».** Es como un bloque de casas con 6 portales diferentes y un **«vecino»** viviendo en cada uno de ellos.

Cada uno de los **«vecinos»** de este conjunto queda identificado perfectamente por dos factores:

— Uno es el nombre *(«número»).*

— Otro es el índice *(o número de la casa en nuestro ejemplo).*

Si hablamos de **«número (2)»,** nos estamos refiriendo al **«vecino»** que vive en la casa 2 del bloque llamado **«número».** El que vive en la casa 5 del bloque sería **«número (5)».** Análogamente podríamos identificar a cada uno de los restantes componentes de la matriz.

Vamos a intentar mejorar el programa 1 mediante el uso de arrays. **¿Quiere intentarlo con nosotros?**

La orden DIM

Lo primero que hay que hacer es «dimensionar» la tabla. En nuestro programa necesitamos espacio para contener los 6 números de la apuesta. Por tanto utilizaremos una línea semejante a:

DIM número (6)

Una vez hecho esto podemos decir que habrá una correspondencia entre las 6 variables que contienen los números aleatorios en el programa 1 y los distintos elementos de la matriz de la siguiente forma:

número 1 = número (1)
número 2 = número (2)
número 3 = número (3)

y así sucesivamente.

De la misma forma comprobaremos si el número obtenido es igual que alguno de los ya calculados y también sacaremos en pantalla los 6 elementos de la matriz una vez que tengamos todos los valores.

Si ha comprendido la «filosofía» de estos nuevos elementos de nuestra programación imaginamos que no le será muy difícil seguir el programa 2.

Primeros pasos

Es bastante más corto que el anterior, **¿verdad?** Vamos a dividirlo en tres partes.

La primera es la comprendida en el bucle que está regido por la variable de control **«indice 1»** (líneas 60 y 120). En ello se obtiene el número aleatorio y si es válido se mete en el elemento de la matriz fijado por **«indice 1».**

La segunda es la que comprueba si es válido el valor obtenido. Para ello recorre la tabla comparando el **«valor»** obtenido con cada uno de los elementos de la misma *(líneas 80 y 100).* Si no es válido, se calcula un nuevo valor y se repite el proceso.

La última es la que nos escribe los resultados. En ella también recorremos la matriz pero esta vez para ir imprimiendo cada uno de sus elementos *(líneas 170 a 200).* Utilizamos como índice la variable **«indice»** a la que hacemos oscilar entre 0 *(primer elemento)* y 5.

Creemos oportuno decirle que nuestro **Amstrad** inicializa todos los elementos de las tablas numéricos con ceros si no le decimos lo contrario. De ahí que la comparación de **«valor»** (línea 90) con los elementos de la tabla que todavía no hemos rellenado nosotros nunca será cierta ya que «valor» no puede valer 0.

Una **«variable dimensionada»** *(por ejemplo «número (1)»)* puede ser utilizada como una variable numérica normal y corriente. Sería correcto teclear dentro de un programa:

```
suma=suma+número (5)
```

o también

```
valor=SIN (número (5)
```

Las matrices alfanuméricas también ayudan

Del mismo modo que podemos agrupar conjuntos de números bajo un mismo nombre, también podemos hacerlo con las variables numéricos o cadenas.

Tecleando:

```
DIM nombre$ (10)
```

reservaríamos en la memoria las posiciones suficientes para contener 10 elementos alfanuméricos bajo la variable «nombre».

Para referirnos a uno de ellos en particular utilizaríamos también un índice. Por ejemplo, si escribimos:

```
PRINT nombre$ (7)
```

aparecería en la pantalla el elemento 7 de la tabla alfanumérica **«nombre».**

Con el sencillo programa 3 obtendríamos un listado con los nombres de los amigos que nos acompañan.

Las líneas 20 a 60 colocan en el elemento de la tabla indicado por la variable **«índice»** cada uno de los nombres que le vamos metiendo. Si éste es la cadena vacía nos salimos del bucle para escribir la lista de las personas que hayan metido su nombre *(líneas 90 a 140)*. La forma de recorrer la matriz es semejante a la anterior.

Una de las cosas más importantes que hay que tener en cuenta es que al dimensionar una matriz de cadena todos sus elementos se inicializan con la cadena vacía. Ojo con ello no nos dé problemas irreparables y nos toque rellenar otra vez la tabla debido a un **«borrado accidental»**.

Las matrices pueden tener más de una dimensión

Vamos a complicar un poco más la cosa. Si le decimos que existen matrices de dos dimensiones, ¿qué cara pondría? No es para tanto, hombre.

Como siempre mantengamos en nuestro pensamiento la idea de agrupar variables de unas características semejantes bajo un mismo nombre. **¿Qué le parece la idea de agrupar en una estructura el nombre, el apellido, la calle y la ciudad donde vive uno de nuestros amigos? ¿Y si probamos a hacer lo mismo con el número de la calle y con el piso?** Todo esto le está haciendo intuir que estamos pensando en crear dos matrices con las siguientes características:

— Una de ellas es alfanumérica y está formada por 100 elementos semejantes compuestos por el nombre, apellido, calle y ciudad de una determinada persona.

— La otra es numérica y también está compuesta por 100 elementos pero ahora cada uno de ellos estará formado por el número de la calle y el piso donde vive esa persona.

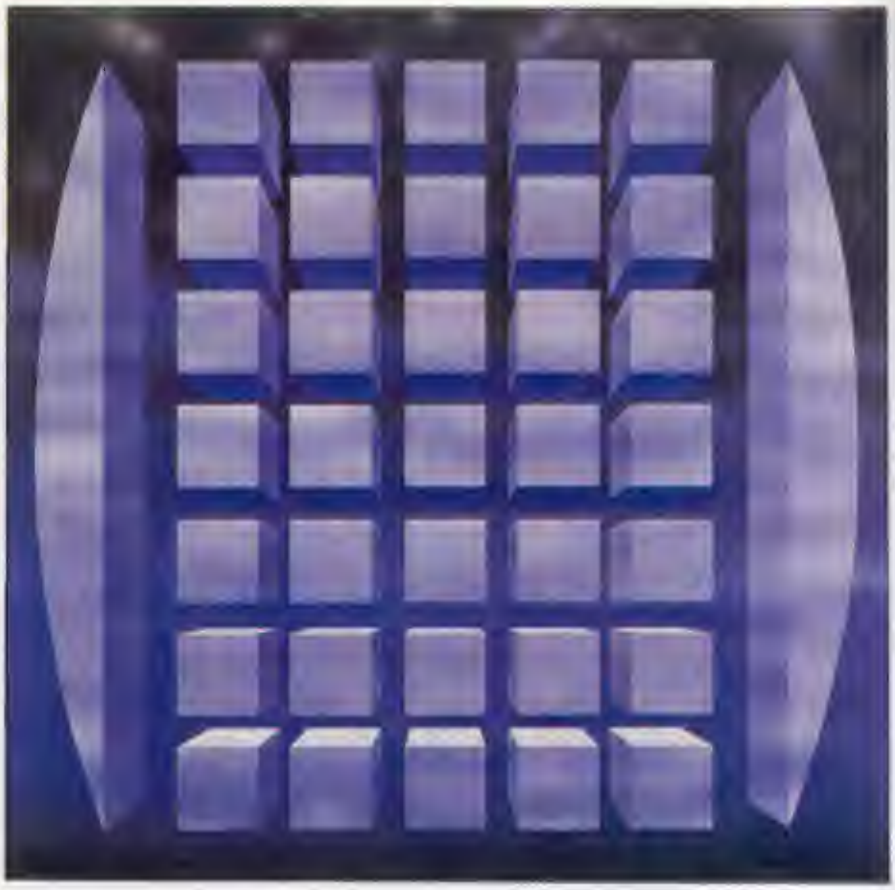
Vamos a reservar espacio en memoria para ellas:

```
DIM nombres$ (100, 4)
DIM número (100, 2)
```

¿Qué significa esto? La primera orden DIM dimensiona una matriz de 100 elementos y cada uno de ellos está dividido en 4 subelementos *(Los cuatro que hemos especificado anteriormente)*.

Para identificar uno de estos subelementos necesitamos fijar dos índices. Con el primero nos colocamos en un **«cajón»** entre 100 donde se guardan todos los datos de una determinada persona. Con el segundo conseguimos situarnos en una **«cajita»**, que hay dentro del **«cajón»** grande, y que ya sólo contiene uno de los datos de la persona elegida.

Por ejemplo, asignamos la **«cajita 1»** al nombre, la 2 al apellido, la 3 al nombre de la calle y la 4 a la localidad donde vive nuestro amigo.



Si accedemos al elemento:

```
nombre$ (2, 4)
```

estaremos colocados en la **«cajita (4)»**, en la que está la localidad donde vive la persona catalogada con el número 2 *(cajón 2)*.

Asignando:

```
apellido$ = nombre$ (20, 1)
```

estaremos colocando el valor del nombre *(cajita 1)* de la persona indicada por el número 20 *(cajón 20)*.

Con la segunda orden DIM ocurre algo parecido. En este caso los valores de los elementos son numéricos pero el sistema de colocarnos en un subelemento particular *(número de la calle o piso)* es semejante al empleado anteriormente.

Si con el segundo índice igual a 1 fijamos el número de la calle y si es igual a 2 estamos en el piso donde vive, con:

```
número (45, 2)
```

será el piso de la casa donde encontraremos a nuestro amigo 45.

Tras toda esta ovalancha de especificaciones teóricas, vamos a intentar hacer una aplicación de estas matrices de dos dimensiones.

Y, ¿por qué no una agenda?

Creemos que una de las primeras cosas que hace todo programador casero es una agenda. Pues bien, vamos a intentar hacerlo nosotros también utilizando las dos matrices bidimensionales que conocemos. **¿Cómo se hará?** El programa 4 nos da la respuesta.

En él rellenamos dos tablas, una alfanumérico y otra numérica, con distintos valores que le vamos dando desde el teclado contestando a las distintas preguntas que nos salen en pantalla.

Cuando ya está lleno o hemos dado un nombre en blanco, el ordenador nos pregunta el número de la persona cuyos datos queremos conocer y coloca el valor que le damos en la variable **«índice»**. Con este valor se coloca en la posición de la tecla *(cajón gordo)* donde se encuentran cada uno de los diferentes datos del mismo amigo *(cajitas pequeñas)*.

Si obtenemos esta **«pequeña filosofía»** de las matrices podremos desarrollarlas de cualquier dimensión (siempre que lo permita la capacidad de nuestro **Amstrad**).

Tome nota de una cosa. Hasta ahora hemos estado diciendo que la instrucción DIM dimensiona un array con un número de elementos igual al valor máximo del índice o al producto de los parámetros entre paréntesis en caso de matrices de varias dimensiones. Ahora le vamos a demostrar que no es así.

Teclee el programa 5 y comprobará usted mismo lo que le decimos:

También está dividido en tres partes. En la primera dimensionamos la matriz (2 * 4) y hacemos igual a cero la variable **«elemento»** *(línea 20 a 40)*.

Es lógico pensar que la matriz tendría 8 ele-

mentos pero si ejecutamos el programa veremos que no. Aunque hayamos dimensionado 8 elementos con la instrucción DIM, el Basic de que está provisto el **Amstrad** hace que dichos elementos empiecen siempre a portir de los índices 0. Esto quiere decir que si hemos dimensionado uno matriz como ésto de la forma:

DIM matriz (2, 4)

no será una matriz de 8 elementos, sino de 15. Habrá que añadirle los elementos siguientes:

matriz (0, 0)
matriz (0, 1)
matriz (0, 2)
matriz (0, 3)
matriz (0, 4)
matriz (1, 0)

y

matriz (2, 0)

Así pues, debido a que los índices comienzon en 0, resultan 15 elementos en lugar de los 8 previstos.

La segunda parte relleno los distintos elementos de la matriz. Asignamos los valores a través de la instrucción READ *(línea 80)* colocada dentro de los bucles FOR... NEXT anidados *(líneas 60-110 y 70-100)*.

La presentación en pantalla de los valores de los elementos se hace mediante otro bucles FOR... NEXT anidados que barren la tabla como yo hemos visto en casos anteriores.

Y una último observoción. El Bosic de nuestro ordenodor permite que no seo necesario dimensionar una lista con la condición de que su índice no sobrepase el valor 10. Asume que se trata de un array de una sola dimensión y con 11 elementos (del 0 al 10).

Una vez que hoyamos utilizodo una matriz en un programa y no la necesitemos más, podemos borrarla de la zona de variables y liberor así el espacio ocupado por ella. La instrucción que realizo esta misión es ERASE.

Supongamos un programa que tenga las siguientes líneas:

10 DIM A (20), B$ (4,3)

y después

1000 ERASE A, B$

La línea 1.000 borrará las variables de matriz A y B$ declaradas en la línea 10. De esta manera el áreo de memoria que ocupaban podró reservarse paro otras finalidades. E incluso, mediante otra sentencia DIM, podremos redefinir otra variable de matriz con el mismo nombre y hasta de distintas características.

¿Qué tol ha ido esta presentación de los variables de matriz? Le aseguramos que en muy poquito tiempo estará en condiciones de mejorar este tipo de variables con gran habilidad. Esto ha sido sólo el comienzo pero poco a poco veremos lo útiles que son y lo forma de tratarlas.

PROGRAMAS

```
10 REM PROGRAMA I
20 MODE 0
30 RANDOMIZE TIME
40 DEF FNaleatorio=INT(RND*49)+1
50 numero1=FNaleatorio
60 numero2=FNaleatorio
70 IF numero1=numero2 THEN GOTO 60
80 numero3=FNaleatorio
90 IF numero1=numero3 THEN GOTO 80
100 IF numero2=numero3 THEN GOTO 80
110 numero4=FNaleatorio
120 IF numero1=numero4 THEN GOTO 11
0
130 IF numero2=numero4 THEN GOTO 11
0
140 IF numero3=numero4 THEN GOTO 11
0
150 numero5=FNaleatorio
160 IF numero1=numero5 THEN GOTO 15
0
170 IF numero2=numero5 THEN GOTO 15
0
180 IF numero3=numero5 THEN GOTO 15
0
190 IF numero4=numero5 THEN GOTO 15
0
200 numero6=FNaleatorio
210 IF numero1=numero6 THEN GOTO 20
0
220 IF numero2=numero6 THEN GOTO 20
0
230 IF numero3=numero6 THEN GOTO 20
0
240 IF numero4=numero6 THEN GOTO 20
0
250 IF numero5=numero6 THEN GOTO 20
0
260 CLS
270 PRINT:PRINT:PRINT
280 PRINT" LOS NUMEROS QUE TE
VAN A HACER        MILLONARIO SON: "
290 PRINT:PRINT
300 PRINT TAB(9)numero1:PRINT
310 PRINT TAB(9)numero2:PRINT
320 PRINT TAB(9)numero3:PRINT
330 PRINT TAB(9)numero4:PRINT
340 PRINT TAB(9)numero5:PRINT
350 PRINT TAB(9)numero6:PRINT
```

```
10 REM PROGRAMA II
20 DIM numero(6)
30 MODE 0
40 RANDOMIZE TIME
50 DEF FNaleatorio=INT(RND*49)+1
60 FOR indice1=0 TO 5
70 valor=FNaleatorio
80 FOR indice2=0 TO 5
90 IF valor=numero(indice2) THEN GO
TO 70
100 NEXT indice2
110 numero(indice1)=valor
120 NEXT indice1
130 CLS
140 PRINT:PRINT:PRINT
150 PRINT" LOS NUMEROS QUE TE
VAN A HACER        MILLONARIO SON: "
160 PRINT:PRINT
170 FOR indice=0 TO 5
180 PRINT TAB(9)numero(indice)
190 PRINT
200 NEXT indice
```

```
10 REM PROGRAMA III
20 DIM nombre$(100)
30 FOR indice=0 TO 100
40 CLS
50 INPUT "COMO TE LLAMAS?   (ENTER
PARA TERMINAR) ",nombre$(indice)
60 IF nombre$(indice)="" THEN GOTO
80
70 NEXT indice
80 CLS
90 PRINT"LISTA DE PERSONAS REUNIDAS
"
100 PRINT"-------------------------
-"
110 FOR indice=0 TO 100
120 IF nombre$(indice)="" THEN END
130 PRINT nombre$(indice)
140 NEXT indice
```

```
10 REM PROGRAMA IV
20 DIM nombre$(100,4)
30 DIM numero$(100,2)
40 REM ENTRADA DE DATOS
50 FOR indice1=1 TO 100
60 CLS
70 INPUT "NOMBRE: ",nombre$(indice1
,1)
80 IF nombre$(indice1,1)="" THEN GO
TO 150
90 INPUT "APELLIDO: ",nombre$(indic
e1,2)
100 INPUT "CALLE: ",nombre$(indice1
,3)
110 INPUT "NUMERO: ",numero(indice1
,1)
120 INPUT "PISO: ",numero(indice1,2
)
130 INPUT "LOCALIDAD: ",nombre$(ind
ice1,4)
140 NEXT indice1
150 REM LISTADO DE DATOS
160 CLS
170 INPUT "DATOS AMIGO NUMERO: ",in
dice
180 IF nombre$(indice,1)="" THEN PR
INT"NO EXISTE.":END
190 PRINT"NOMBRE: "
200 PRINT nombre$(indice,1)+" "+nom
bre$(indice,2)
210 PRINT"CALLE: ";nombre$(indice,3
);" NUMERO: ";numero(indice,1)
220 PRINT"PISO: ";numero(indice,2)
230 PRINT nombre$(indice,4)
```

```
10 REM PROGRAMA V
20 REM INICIALIZAR
30 DIM matriz(2,4)
40 elemento=0
50 REM DAR VALOR A LOS ELEMENTOS
60 FOR indice1=0 TO 2
70 FOR indice2=0 TO 4
80 READ valor
90 matriz(indice1,indice2)=valor
100 NEXT indice2
110 NEXT indice1
120 REM LISTADO VALORES
130 CLS
140 FOR indice1=0 TO 2
150 FOR indice2=0 TO 4
160 PRINT"MATRIZ(";indice1;",";indi
ce2;") =";matriz(indice1,indice2)
170 elemento=elemento+1
180 NEXT indice2
190 NEXT indice1
200 PRINT
210 PRINT"LA MATRIZ TIENE";elemento
;"ELEMENTOS"
220 DATA 1,2,3,4,5,6,7,8,9,10,11,12
,13,14,15
```

Los secretos del teclado son así

Habíamos dejado el tema en la definición de cualquier tecla. En el presente artículo se intenta destripar la memoria de su CPC para entener algo más esta nueva herramienta. Se ofrece un programa para ingresar en la nueva era del teclado, el teclado DVORAK. Anticípese con él, a un futuro cada vez más próximo.

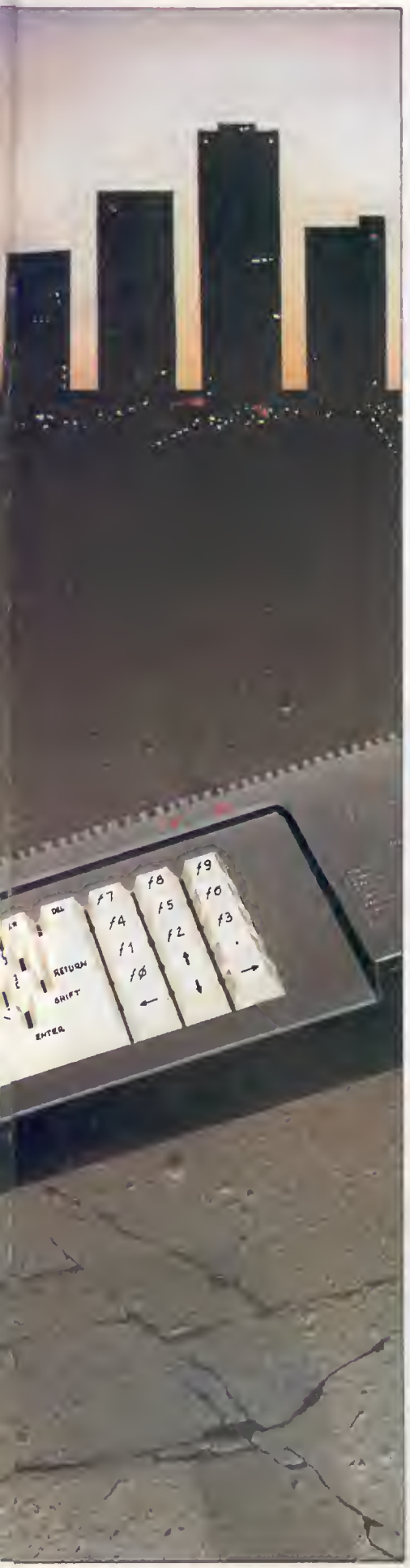
En el artículo anterior afirmábamos que sólo eran asignables 32 comandos expandibles, no obstante, esto no significa que únicamente podamos definir 32 teclas distintas. Podemos naturalmente, asignar a varias teclas el mismo código, aunque esto, difícilmente le servirá para algo. La otra posibilidad, en la cual nos vamos a basar en el programa 2, consiste en asignar a las teclas un código no expandible, es decir, ya definido, lo cual equivale a asignarle un código ASCI. Contrariamente a lo que pudiéramos pensar el teclado se encuentra totalmente definido en RAM y no en ROM. El programa 1 permite echarle un vistazo a estos valores, pruebe a casar la numeración del teclado con el código ASCI de su contenido. El teclado se encuentra ubicado normalmente a partir de la posición 46230. Ahora teclee por ejemplo:

KEY DEF 69,1,49,50,51

Ahora si pulsamos la tecla A obtendremos, 1, 2 ó 3 según la pulsemos sola o acompañada de SHIFT o de CTRL respectivamente. Recordemos que 49,50 y 51 son los códigos ASCI de 1, 2 y 3. Ejecute nuevamente el programa 1, y mire los cambios ocurridos.

El programa 2 aprovecha esta posibilidad para deshacer el viejo teclado QWERTY y convertirlo en el DVORAK. En el teclado QWERTY, teclado que actualmente utiliza cualquier ordenador o máquina de escribir, la disposición de las teclas se eligió de forma que fuese un inconveniente para conseguir grandes velocidades. La razón estaba en que aquellas máquinas no permitían escribir a cualquier velocidad, sobrepasado el límite la máquina se atascaba, con lo que se perdía más tiempo que escribiendo un poco más despacio. El modelo de teclado QWERTY fue adoptado por todos los fabricantes imponiéndose en prácticamente todo el mundo, sin embargo hoy día, en la era de la electrónica esta limitación se ha quedado obsoleta, razón por la que en Estados Unidos se ha investigado en un nuevo teclado que no sólo no pondrá freno a los dedos del mecanógrafo, sino que lo incitará a una escritura más veloz. En la figura 1 se ofrece cómo queda el teclado tras la ejecución del programa 2. Utilice este nuevo teclado y vaya preparándose para el futuro.

Sin embargo, como son muchos los años que al teclado QWERTY le quedan por delante, el programa 3 lo que hace es convertir el QWERTY sajón en el QWERTY español, incluyendo la # al lado de la 1. La tecla TAB contiene el por (*) y los dos puntos (:).



Caracteres de control para el editor

El editor no es sino un programa que ejecutándose en su CPC, le permite escribir en pantalla un conjunto de caracteres. El control del editor termina en cuanto comienza a ejecutarse un programa Basic, CM... Sin embargo, el programa cuando solicite una entrada devolverá el control nuevamente al editor. En el artículo anterior comprobábamos cómo los códigos de control usuales no provocaban el esperado efecto para el editor. Estos sólo eran entendidos en la forma de códigos de control por el Basic.

Si observa el programa 1 con atención, la tecla 0, tecla de cursor arriba contenía el caracter «^», correspondiente al caracter número 240, lo que sucede es que el editor realiza un chequeo y cuando intentamos enviar este caracter a la pantalla, lo que hace realmente es subir el cursor una línea. A continuación damos una lista de los caracteres que el editor reconoce como caracteres de control:

Código ASCI	Efecto
253	Conecta/desconecta el SHIFT (CAPS LOCK).
224	Copia el caracter que se encuentra en el segundo cursor.
127	Hace retroceder el cursor y borra el caracter que encuentra.
13	Enter.
252	Provoca un BREAK.
	Borra el caracter que se encuentra en la posición del cursor.
240	Cursor arriba.
241	Cursor abajo.
242	Cursor izquierda.
243	Cursor derecha.

La aplicación de estos controles en la definición de nuestras teclas puede resultar de una potencia asombrosa.

Por ejemplo si desea anular la tecla BREAK de su programa bastaría con incluir en él:

KEY DEF 66,1,0,0,0

Lo que hemos hecho ha sido asignar a la tecla 66 el caracter nulo (0), que equivaldría a un no hagas nada, para las tres formas de pulsarla sola con SHIFT o con control. Incluso ahora podríamos camuflar una teclo BREAK en el teclado. Teclee:

KEY DEF 36,0,252,252,252

Ahora la tecla L será la que provoque un BREAK.

Existen programas que si se juega con el cursor mientras se está ejecutando éste, podemos perder toda la estética del diseño de la pantalla, una buena forma de remediar este problema es inhibir los cursores, sabría hacerlo por sí mismo.

Realmente mediante estos códigos se pueden lograr verdaderas maravillas, aumentando las posibilidades del editor. El pragrama 3 convierte a las teclas f2 y f3 en una herramienta potente para la carga de programas de disco. La tecla f2 se convierte en el comando CAT, mostrar directorio, y la tecla f3 ordena cargar el programa que se encuentra bajo el segundo cursor. Un ejemplo práctico de su utilización sería:

Pulsamos f2 y obtenemos el directorio en pantalla.

Ahora manteniendo pulsada la tecla SHIFT movemos el cursor hasta el nombre del programa que queremos cargar, una vez hecho esto pulsamos f3 y el programa se cargará automáticamente.

El programa 4 lo que hace es habilitar la tecla TAB para copiar 40 caracteres, una línea completa en el modo de pantalla 1.

Observe cómo para la definición de estas teclas nos vemos en la necesidad de utilizar códigos expandibles ya que deseamos que las teclas contengan más de un caracter. Se ha empleado un pequeño truco para llenarlas, el uso de un bucle For-next, esto no debiera de incomodarle excesivamente ya que únicamente se ha hecho con el objetivo de evitar teclear repetidamente CHR$(X).

Por si aún no lo había observado se puede resaltar que aunque antes no los hemos citado como tales, los códigos entre el 128 y el 159 son interpretados también como códigos de control por el editor. Lo que hace el editor cuando encuentra un valor comprendido en este intervalo es acudir a ese comando expansible para presentarlo en pantalla.

Donde N,M... corresponden a los códigos que hacen que la impresora escriba en negrita, élite, doble posada... Para el conocimiento de estos códigos consulte el manual de su impresora. De este modo podrá seleccionar el modo de impresión mediante la pulsación de una sola tecla.

También la puede utilizar para que el operador pueda dar respuesta automática, a las preguntas que su programa vaya realizando lográndose de este modo prever errores, como que el operador responda con «**s**» cuando esperaba un «**SI**» o un «**NO**»; o con «**si**» o «**no**» cuando esperaba un «**SI**» o un «**NO**». El programa 6 pone un ejemplo de esta aplicoción.

Como última broma el programa 7 el cual deberá saber lo que hace, sin necesidad de ejecutarlo.

A fin de cuentas binario

Mediante el programa 2 lo que conseguíamos mostrar era donde se encontraba definido el teclado. Los comandos expandidos por el usuario también tienen una porte reservada parte de la memoria. El comienzo de esta zona viene dado como lo que se conoce como una variable del sistema, lo dirección de inicio se corresponde con el valor.

PEEK(&B62B)*256+PEEK(&B62C)

El programa 5 realiza la misma labor que el programa 1, observe cómo cada una de las frases que contienen nuestros tokens están separadas por un número que expresan la longitud de éstos, para los valores menores a 32 no mandamos imprimir el caracter correspondiente pues esto no haría sino desorganizarnos la pantalla por ser estos caracteres de control paro el Basic.

Ya que conocemos la dirección de inicio de estos códigos nos bastará con salvar este trozo de memoria en la cinta o en el disco para que tengamos ampliado el teclado.

Realizemos un ejemplo, ejecute el programa 5, y ahora teclee:

SAVE «**TECLAS**»,B,PEEK(&B62B)*(&B62C),140

Pongo en marcha el cassette si posee un 464, y espere a que toda la traza de memorio se solve. Ahora haga un reset, o mejor aún si no se fía desenchufe su ordenador. Vuélvalo a conectar y haga:

LOAD «**TECLAS**»

Una vez que reciba el informe READY, señal de que se ho volcado el contenido del fichero en la memoria de su CPC, pulse f1 o f2 o f3... Ahí están de nuevo.

Para terminar algunas aplicaciones

El partido que a todo lo expuesto a lo largo de los dos artículos pueda sacor, es cosa suyo y de sus necesidades depende. Puede por ejemplo, si tiene la fortuna de disponer de impresora, definir las teclas de función como:

KEY N,"**PRINT** 8,CHR$(N);CHR$(M).."+CHR$(13)

ProgramAcción

```
10 REM
20 MODE 2
30 WINDOW #1,22,58,2,23:PAPER #1,1:
PEN #1,0:CLS#1
40 LOCATE 22,1:PRINT " PEEK
ASCI        COD      CHR$"
50 FOR n=46230 TO 464856
60 PRINT #1,TAB (1);n;TAB (13);PEEK
(n);TAB (25);m;:IF PEEK(n)>32 THEN
PRINT #1,TAB(34);CHR$(PEEK(N))
70 m=m+1
80 'PRINT#1
90 NEXT n
```

```
10 REM
20 REM
30 SYMBOL AFTER 254
40 SYMBOL 254,254,0,198,230,254,206
,198,0
50 SYMBOL 255,126,0,216,102,102,102
,102,0
60 KEY 130,CHR$(254)
70 KEY 129,CHR$(255)
80 KEY DEF 29,1,129,130
90 KEY DEF 68,1,58,42
```

```
10 REM
11 REM
20 REM
30 a$=STRING$(40,224)
40 KEY 1,a$
```

```
10 REM
20 REM
30 KEY 159,"esta tecla es la mia"
40 MODE 2
50 KEY DEF 15,1,74
60 lon=PEEK (&B62A)
70 inicio=PEEK(&B62C)*256+PEEK(&B67
B)
80 FOR n=inicio TO inicio +lon
90 PRINT n,PEEK(n),:IF PEEK(n)>32 T
HEN PRINT CHR$(PEEK(n));
100 PRINT
110 NEXT n
120 J
```

```
10 REM
---
20 REM
30 a$=CHR$(65)+CHR$(68)+CHR$(77)+CH
R$(79)+CHR$(83)
40 KEY 128," "+a$+" "
50 FOR n=0 TO 71
60 KEY DEF n,1,128,128,128
70 NEXT
```

INTRODUCCION AL CP/M

En el mundo de los microcomputadores de 8 bits susceptibles de uso profesional y técnico, además del meramente lúdico, las palabras sistema operativo tienen una sonoridad constante y una importancia fundamental. CP/M es el rey indiscutido de los sistemas operativos para ordenadores de 8 bits basados en el microprocesador Z80 y similares, como su Amstrad.

CP/M son las siglas de **Control Program Monitor,** y, como su nombre indica, se trata de un programa capaz de controlar el hardware de un computador al nivel más bajo posible, a nivel de lenguaje ensamblador, actuando de intermediario entre lo **físico** del ordenador, otros programas y usted. A este tipo de programa se le conoce como **sistema operativo,** en inglés Operating System *(O.S.)*. Cuando el O.S. está basado en ficheros de disco y es capaz de manejarlos, se le llama ingeniosamente Disc Operating System, DOS para los amigos.

Después de una presentación tan aburrido, bueno, pues CP/M es un DOS y su historia resulta extremadamente curiosa.

¿Cómo nació CP/M?

Yo creo que es un claro ejemplo del axioma que parece funcionar muy bien en informática: **«el que da primero, da dos veces.**

Hace muchos, muchos años, en la Edad del Bronce de la informática, en 1973, el doctor Gary Kildall desarrolló CP/M.

La primera versión fue creada para el propio sistema experimental de Kildall, el cual contaba con una de las primeras unidades de discos de 8 pulgadas fabricada por Shugart Associates *(la hoy famosa «mecánica Shugart»)*.

A Kildall no le quedó otra que sacarse un DOS decente prácticamente de la manga para poder trabajar con su equipo, ya que por aquella época uno recibía el hardware de los ordenadores completamente desnudo a poco que se descuidara y tenía que resolvérselo como podía, ensamblador en ristre.

El padre de CM/M, además de ser un buen programador, también era consultor de software de Intel, y les mostró las primeras versiones de su sistema operativo.

Intel se desentendió del tema y renunció a comercializar y desarrollar el proyecto. Tal vez esto parezca sorprendente, porque hoy en día tenemos muy claro la importancia de un sistema operativo cuasi-estándar e independiente del hardware del ordenador sobre el que corra, pero en 1973 muy poca gente poseía computadores y los que lo poseían, particulares sobre todo, no estaban muy seguros de qué hacer con ellos.

Hacia 1975, muy pocas compañías vendían ordenadores y las pioneras en este campo, como Altair, Polymorphic y Processor Technology fabricaban sus propios sistemas operativos en los sistemas que incluían discos. CP/M no hubiera sobrevivido si estas firmas se hubieran espabilado un poco más a la hora de proporcionar productos de software a los consumidores.

Algunas otras firmas, como Tarbell Electronics y Digital Microsystems, tal vez con más astucia o más ambición decidieron abaratar costos en investigación y desarrollo de DOS, adoptando CP/M para sus productos.

Por otra parte, estas compañías vendían componentes que podían instalarse virtualmente sobre cualquier equipo, por lo que usuarios de Altair, Vector y otros no tuvieron que aguardar a que la firma a la que habían comprado el ordenador desarrollara sistema operativo alguno; adoptaron el CP/M y listo.

Otros fabricantes siguieron el ejemplo de IMSAI, que distribuía sistemas de disco sin software para usarlos y había prometido paliar rápidamente el problema suministrando un sistema operativo adecuado, que se llamó IMDOS y era un CP/M con todas las de la ley, sólo que disfrazado un poco, por decoro, supongo.

No hay que olvidar en ningún momento quiénes fueron los primeros usuarios de CP/M y lo decisivamente que contribuyeron al auge de la obra de Kildall.

Los pioneros

Estas personas eran auténticos aficionados, locos por los ordenadores y en su camino de investigación encontraban fre-

cuentemente obstáculos insuperables debido a la escasez de equipos y el caos de incompatibilidades que reinaba entre ellos. Por eso, CP/M les vino como agua de mayo, ya que permitía enlazar cualquier micracomputadar basado en el Z80 o en el 8080 con cualquier sistema de discos. Apareció, por tanto, un grupo de usuarios con sistemas mixtos muy fuerte, muy visible y, sobre todo, con grandes ganas de ampliar sus ya considerables conocimientos informáticas.

Después de esto, lo que podríamos llamar las primeros balbuceos de **CP/M**, vino en su ayuda el avance tecnológico: comenzaron a aparecer unidades de disco fiables.

Programas transportables

Sólo faltaba un punto, que esta vez vino del software, para que la informática se convirtiera en un negocio a escala mundial: escribir programas que pudieran ejecutarse sobre la mayor cantidad de ordenadores posible, sin ningún cambio, es decir, que bastará con implementarlos una sola vez y se pudieran vender muchas veces.

Esto fue el espaldarazo final de CP/M, porque era uno de las pocos DOS que podrío implantarse en cualquier computador y no se restringía a una clase determinada de discos.

De todas maneras, CP/M también contó con algo de suerte mezclada con lógica: los primeros programas que aparecieron bajo este sistema operativa fueron herramientas de desarrollo, esto es, programas que permitían al programador generar otros programas, como por ejemplo EBASIC, CBASIC, Microsoft BASIC y otros programas especiales en lenguaje ensamblador.

Con estas herramientas se produjeron programas orientados al usuario final, o sea, al que programar le importaba un bledo y sólo quería algo hecha que le facilitara el trabajo, como por ejemplo programas de contabilidad, bases de datos, procesadores de textos y el larguísimo etc. que hoy puede verse en cualquier catálogo CP/M.

La escalada de CP/M

Esta situación originó un caso clínico de **«pescadilla que se muerde la cola»**, que ni siquiera hoy, en 1986, muestra signos de detenerse: CP/M ayudaba a la proliferación de lenguajes de programación y herramientas de desarrollo que a su vez impulsaban la aparición de más programas de aplicación que a su vez ampliaban el círculo de usuarios de CP/M, creando la necesidad de más y más aplicaciones y así **«ad infinitum»**.

Llegamos a otro punto clave de la historia de CP/M y de la informática en ge-

neral: el Año del Señor de 1981, punto de inflexión para los ordenadores por el hecho de que las grandes compañías como IBM, Hewlett Packard y Xerox, se bajaron del pedestal de los mainframes y se nos introdujeran, calladamente y en silencio, hasta en la sopa, como aquel que dice.

CP/M formó parte enseguida de la fiesta, porque algunos de los nuevos equipos la ofrecieron como sistema operativa principal y prácticamente todos como opcional. El número de usuarios pasó del cuarto de millón en ese año.

Hoy por hoy, más de 300 firmas de ordenadores poseen CP/M para sus equipos, bien sea CP/M-80 o CP/M-86 y, encima, los propietarios de máquinas con microprocesadores diferentes del 8080, Z80, 8080 y 8086 pueden ejecutarlo mediante tarjetas que llevan incorporado un Z80 *(caso del Apple)*, mientras que otros, como Commodore, lanzan sus máquinas al mercado dotadas de más de un microprocesador, el **propio** y un Z80 para correr CP/M *(caso del Commodore 128)*.

Hace no mucho tiempo, otra marca de ordenadores se incorporó al carro de CP/M, si bien de una forma un tanto peculiar: se trata de **Amstrad,** de su **Amstrad** y ahora entra usted directamente en escena.

Hemos dicho un poco más arriba que una de las mayores excelencias de CP/M en particular y de cualquier sistema operativo que se precie en general, es la independencia respecto al Hardware de la máquina.

¿Compatibles todos?

Bien, esto es cierto para CP/M, pero... dentro de unos límites. Hay una parte de CP/M llamada BIOS *(Basic Input/Output System)* que sí depende del hardware y debe ser adaptada por el fabricante o por el usuario a una máquina particular.

Usted no puede coger un CP/M , sin más ni más y meterlo al **Amstrad** directamente.

No es que esto importe mucho, ya que la casa **Amstrad** se encarga de resolvérnoslo, pero hay que matizar lo que significa compatibilidad, aun dentro del mismo sistema operativo.

Otra peculiaridad de **Amstrad** es el formato escogido para sus unidades de disco: las tres pulgadas están muy lejos de ser un estándar como los discos de 5 1/4 pulgadas; por ello, cuando se habla de **miles de programas en CP/M** para un ordenador, hay que tener presente primero si existen en el formato de discos de su máquina. Si no es así, o usted o alguien tendrán que adaptarlos, y si uno no está por la labor, los miles de programas se ven reducidos a unos sumarios «cientos» o «decenas».

Los **Amstrad,** como nuestros lectores saben muy bien, emplean dos versiones de CP/M, la 2.2 (CPC464, CPC664) y la 3.0 *(CPC6128, PCW8256)*.

CP/M para más de 64 K

Ambas son muy diferentes y están pensadas para cosas distintas también. La versión 2.2 es el CP/M «de toda la vida», capaz de gestionar un máximo de 64 Kbytes de memoria, mientras que la versión 3.0 y siguientes, conocidas como CP/M Plus, pueden gestionar más de 64 K mediante la técnica de paginación de memoria (128 K en el 6128 y 256 K en el 8256).

El estudio detallado de ambas versiones sobrepasa el alcance de este artículo, pero sí podemos decir que CP/M Plus es mucho más potente que CP/M 2.2 y responde a un concepto más avanzado en cuanto a la gestión de los recursos de un ordenador y, por supuesto, en lo referente a la facilidad de uso *(se nota la memoria extra)*.

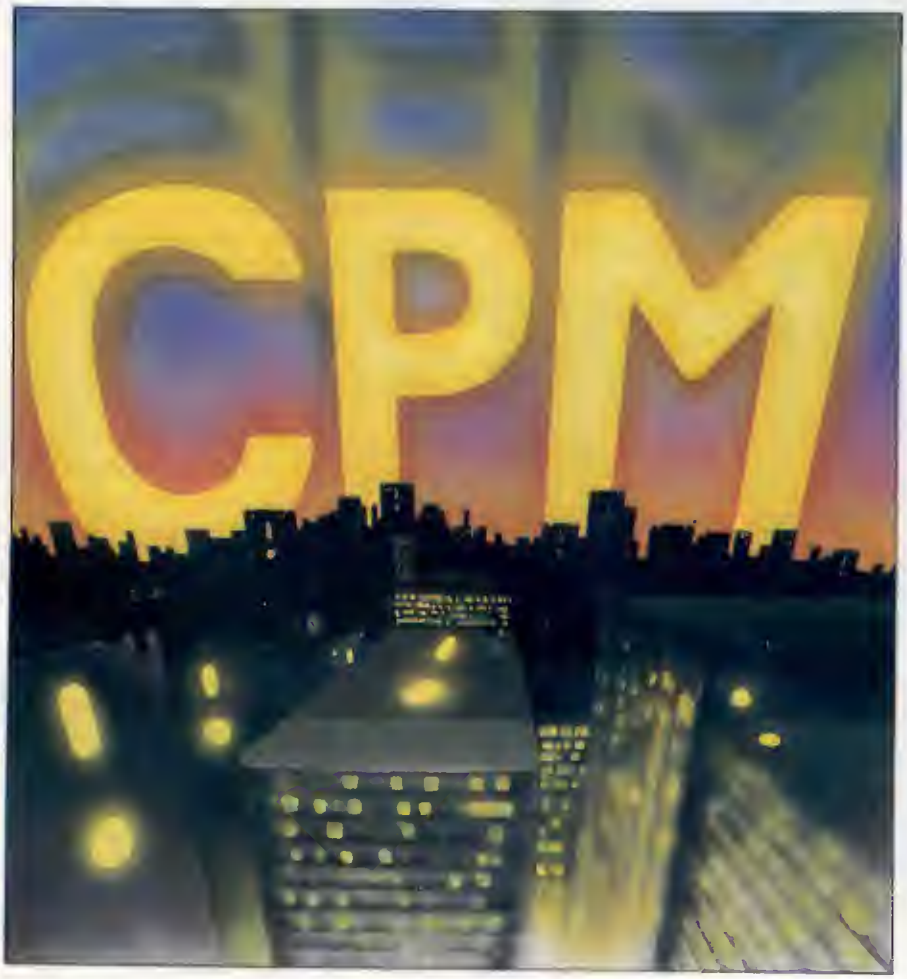
Ya puestos a decir, no estaría de más recordar que CP/M es un sistema operativo orientado a comandos, esto es, usted se comunica con el sistema mediante órdenes preestablecidas residentes en el disco del sistema.

Por razones de ahorro de memoria, que se remontan a épocas anteriores de la informática, CP/M divide sus órdenes en dos tipos: las más comunes, permanecen continuamente en memoria central; son las llamadas «órdenes residentes». Las menos comunes, que incluyen el resto de las órdenes CP/M y los programas de aplicación, se denominan **órdenes transitorias,** y son invocadas del disco cuando son necesarias. Una vez que su función ha terminado, son olvidadas por el ordenador y tratadas como **garbage** *(basura)*.

Normalmente, los nombres de ficheros CP/M constan de un máximo de 8 caracteres más un máximo de tres para el apellido; para las órdenes del sistema operativo, el apellido elegido es **COM,** de COMando *(ver gráficos 1 y 2)*.

Respecto al CP/M Plus, los nombres son consistentes con la versión 2.2, para mantener la compatibilidad, pero la versión 3.0 cuenta con muchos más comandos y utilidades, junto con algo muy importante que la 2.2 no tiene: la extensión GSX *(Graphics System Extension)*, que permite a cualquier programa que corra bajo CP/M Plus hacer pleno uso de los recursos gráficos del ordenador en cuestión y manejar impresoras y plotters.

En fin, esperamos poder abordar pronto con el mayor detalle, tanto desde el punto de vista del usuario como del programador, que es el CP/M byte a byte y que puede hacerse con él. De momento, esperamos igualmente que este breve artículo haya cumplido su propósito: contar la historia de CP/M e introducirnos suavemente en el mundillo que encarna en él, tan distinto del entorno del Basic.



Gráfica 1

Directorio del disco del sistema CP/M 2.2.

```
dir
A: MOVCPM  COM : PIP      COM : SUBMIT   COM : XSUB     COM
A: ED      COM : ASM      COM : DDT      COM : LOAD     COM
A: STAT    COM : DUMP     COM : DUMP     ASM : AMSDOS   COM
A: FILECOPY COM : SYSGEN  COM : BOOTGEN  COM : COPYDISC COM
A: CHKDISC COM : DISCCOPY COM : DISCCHK  COM : SETUP    COM
A: FORMAT  COM : CSAVE    COM : CLOAD    COM : EX1      BAS
A: EX2     BAS : ROINTIME DEM : RITDEMO  BIN : DISC     BAS
```

Gráfica 2

Ordenes transitorias de CP/M 2.2.

```
A>dir *.com
A: MOVCPM  COM : PIP      COM : SUBMIT   COM : XSUB     COM
A: ED      COM : ASM      COM : DDT      COM : LOAD     COM
A: STAT    COM : DUMP     COM : AMSDOS   COM : FILECOPY COM
A: SYSGEN  COM : BOOTGEN  COM : COPYDISC COM : CHKDISC  COM
A: DISCCOPY COM : DISCCHK COM : SETUP    COM : FORMAT   COM
A: CSAVE   COM : CLOAD    COM
```

Gráfica 3

Disco de sistema CP/M 3.0 CARA 1.

```
dir
A: DISCPM  EMS : BANKMAN  BAS : PROFILE  ENG : SUBMIT   COM
A: SETKEYS COM : KEYS     CCP : LANGUAGE COM : SET24X80 COM
A: PALETTE COM : SETSIO   COM : SETLST   COM : DISCKITS COM
A: DATE    COM : DEVICE   COM : DIR      COM : ED       COM
A: ERASE   COM : GET      COM : PIP      COM : PUT      COM
A: RENAME  COM : SHOW     COM : TYPE     COM : SET      COM
A: SETDEF  COM : AMSDOS   COM : BANKMAN  BIN : KEYS     WP
```

Ordenes CP/M 3.0 CARA 1.

```
A>dir *.com
A: SUBMIT  COM : SETKEYS  COM : LANGUAGE COM : SET24X80 COM
A: PALETTE COM : SETSIO   COM : SETLST   COM : DISCKITS COM
A: DATE    COM : DEVICE   COM : DIR      COM : ED       COM
A: ERASE   COM : GET      COM : PIP      COM : PUT      COM
A: RENAME  COM : SHOW     COM : TYPE     COM : SET      COM
A: SETDEF  COM : AMSDOS   COM
```

```
A> c
```

Gráfico 4

Disco de Sistema CP/M 3.0 CARA 2.

```
A dir
A: AMSDOS  COM : SID      COM : DUMP     COM : GENCOM   COM
A: HEXCOM  COM : HIST     UTL : INITDIR  COM : LIB      COM
A: LINK    COM : MAC      COM : PATCH    COM : RMAC     COM
A: SAVE    COM : TRACE    UTL : XREF     COM : DD-DMP1  PRL
A: DDSHINWA PRL : DDHP747? PRL : ASM      COM
```

Ordenes CP/M 3.0 CARA 2

```
A>dir *.com
A: AMSDOS  COM : SID      COM : DUMP     COM : GENCOM   COM
A: HEXCOM  COM : INITDIR  COM : LIB      COM : LINK     COM
A: MAC     COM : PATCH    COM : RMAC     COM : SAVE     COM
A: XREF    COM : ASM      COM
```

Gráfico 5

Disco D.R. LOGO CP/M 3.0

```
A>dir
A: AMSDOS  COM : HELP     COM : HELP     HLP : SUBMIT   COM
A: SETKEYS COM : KEYS     CCP : KEYS     DRL : LOGO3    SUB
A: GSX     SYS : GENGRAF  COM : DDMODE2  PRL : DDFXLR7  PRL
A: DDMODE1 PRL : DDMODE0  PRL : ASSIGN   SYS : DRIVERS  GSX
A: LOGO3   COM
```

Ordenes CP/M 3.0

```
A>dir *.com
A: AMSDOS  COM : HELP     COM : SUBMIT   COM : SETKEYS  COM
A: GENGRAF COM : LOGO3    COM
```

HIGHWAY ENCOUNTER

La columna de incansables robotrones, camina en prefecta formación por la autopista de la muerte. Programados para cumplir su misión, uno tras otro van cayendo, impasibles los restantes compañeros continúan su marcha hacia la base Alfa.

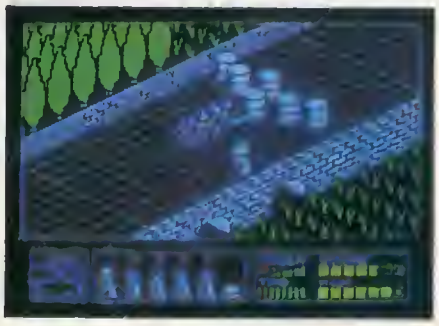
Todo empezó aquel día en el que el supremo Simonsen, tomó la tajante decisión de reducir a cenizas la base Alfa.

Esta base, es la sede de los terribles devoradores de chatarra, mostruos insaciables, con formas dantescas que vigilan el camino de acceso a la misma.

Esta senda, conocida como la autopista de la muerte es una vía totalmente rectilínea, que une las tierras bajas de Gorf con las altas planicies de Kemston, donde las plantas y los árboles han sido exterminados para depositar en su lugar, los elementos mecánicos que produce la inalcanzable base.

En sus inmediaciones, está la plantación de ojos mecánicos, dispuestos a ser activados en cualquier momento.

Compatible: CPC/464, CPC/664 y CPC/6128.



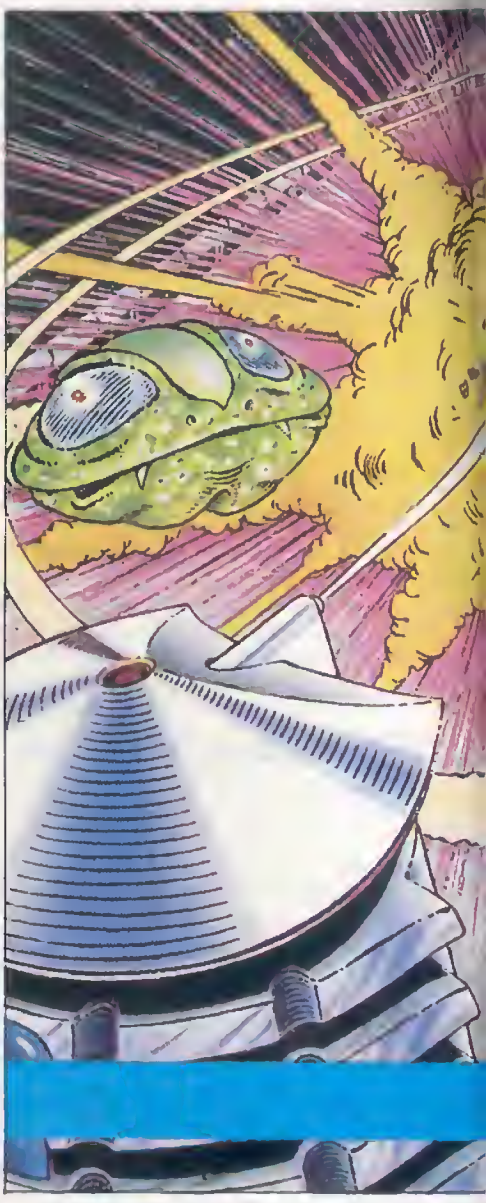
Simonsen el grande ha tomado esta tajante decisión, porque ve incrementarse día a día el poderío y la producción de ojos mecánicos de la factoría, temiendo una inminente invasión.

Ante estas circustancias extremas, decide dar un golpe de mano, antes de que sea demasiado tarde y su reino sea invadido y transformado en otra factoría de ojos.

Simonsen, para llevar a cabo su plan, selecciona a los cinco ejemplares más perfectos de su factoría de robotrones, botes que se encargan de mantener con vida al país y abastecer el palacio presidencial.

Para su importante misión, los programadores hacen un esfuerzo especial y modifican las tarjetas de operación de los mismos.

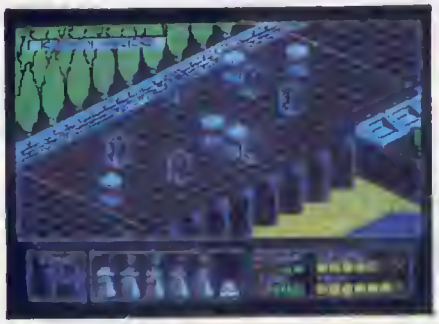
El nuevo programa sólo dice avanzar o morir y está estructurado con la ferrea disciplina militar.



Guardando estricta formación, los robotrones deben avanzar por la pista; en cabeza el cabo robotrón y detrás en columna el resto de la tropa.

Todo el desgaste y el peso de la lucha, lo lleva el cabo, éste combatirá hasta la muerte, mientras tanto la tropa viaja detrás, cubiertas del fuego enemigo por el cabeza de formación, esperando el momento de entrar en acción cuando éste caiga.

Hasta ese momento, sólo se ocu-



Mr. Joystick

y empuje, la columna se dirige hacia el primer escoyo, la muralla de bidones, nuestros botes se abren paso entre ellos y encuentran al primer ojo mecánico.

En las siguientes zonas, nos movemos de sorpresa en sorpresa, cada una posee sus propios peligros y moradores peculiares.

Las antorchas letales, prismas cristalinos, columnas vulnerables a nuestros disparos, prismas pétreos, estrellas de la muerte, robots boca y platillos flotadores, son algunos de los peligros y objetos que nos pueden aniquilar, o de los cuales podemos sacar algún partido en nuestro beneficio.

Llegar hasta la base Alfa no es cosa fácil, en el camino hacia ella muchos de nuestros androides caerán, e incluso nos quedaremos a la puerta de la misma, sin poder alcanzar nuestro objetivo.

En esta ocasión Vortex, nos deleita con uno de sus maravillosos juegos.

Los que hayan tenido contacto con el spectrum, quizás hayan oído hablar del Tornado Low Level, o más posteriormente del fabuloso ciclón.

Todos los productos de esta casa, se caracterizan por el uso en los gráficos, de la representación plana tridimensional.

En este campo, Vortex ha conseguido verdaderos logros, extendiendo hasta límites insospechados, las posibilidades del spectrum.

Highway enconunter, mejora notablemente los resultados anteriores de la casa, consiguiendo unos gráficos claros, con una animación suave y muy bien conseguida.

El decorado de las distintas pantallas, está realizado con un cuidado exquisito, transportándonos a una era mecánica, donde los robots y distintos seres mecánicos, son protagonistas principales.

Descubrir nuevas pantallas, según avanzamos hacia la fatídica zona cero es una completa gozada.

Un digno exponente de la nueva generación de software.

pan de empujar la tetrabomba que deben depositar en la base.

Introducidas las tarjetas de proceso, los robotrones son transportados hacia el final de la autopista y abandonados a su suerte.

Ahora todo depende de su fuerza

Es usted capaz de tomar el relevo del General Montgomery...?

& Juegos ESTRATEGIA

le presenta en exclusiva el WAR GAME, para Spectrum, de mayor éxito en Inglaterra:

ARNHEM

(operación «Market Garden», basada en un hecho real de la Segunda Guerra Mundial)

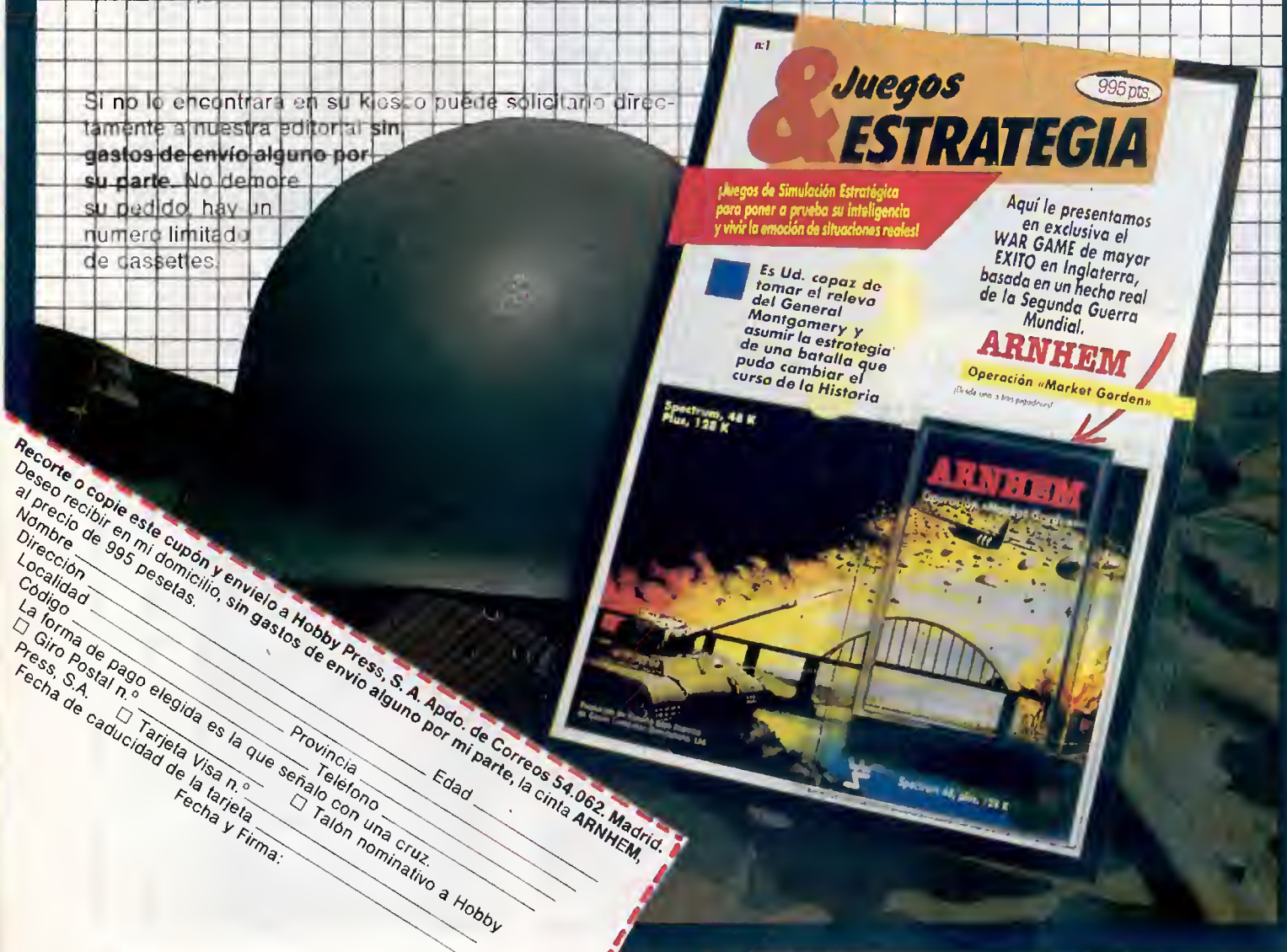
Ya está a la venta!

Y ahora también para

AMSTRAD

Compatible todos los modelos

Si no lo encontrara en su kiosco puede solicitarlo directamente a nuestra editorial **sin gastos de envío alguno por su parte.** No demore su pedido, hay un número limitado de cassettes.



Recorte o copie este cupón y envíelo a Hobby Press, S. A. Apdo. de Correos 54.062. Madrid. Deseo recibir en mi domicilio, sin gastos de envío alguno por mi parte, la cinta **ARNHEM,** al precio de 995 pesetas.

Nombre _____ Edad _____

Dirección _____ Teléfono _____

Localidad _____ Provincia _____

Código _____

La forma de pago elegida es la que señalo con una cruz.

☐ Giro Postal n.º _____ ☐ Tarjeta Visa n.º _____ ☐ Talón nominativo a Hobby Press, S.A.

Fecha de caducidad de la tarjeta _____

Fecha y Firma: _____

HOBBY PRESS, S.A. Editamos para gente inquieta.

Cambio de Colores

Análisis

Las largas horas de trabajo ante la pantalla de un ordenador, nos obligan a plantearnos la pregunta: ¿Cuáles son los colores más idóneos para la diferenciación de la tinta y el papel?

Seguramente muchos usuarios, estén ya muy hartos del azul marino y el amarillo brillante.

Esta vez Análisis, desmenuza un pequeño programa que nos ayudará a elegir los colores de tinta y papel, más de acuerdo con nuestro gusto, con un toque de tecla.

10-30 Cabecero del listado.

40 Limpia la pantalla.

50-90 Líneas que imprimen el texto, que nos va a ayudar a distinguir la tinta del fondo.

60 Toma como inicio de la impresión en pantalla, la línea quinta.

70-90 Ciclo FOR... Next, que imprime en pantalla el mensaje Microhobby **Amstrad** semanal.

110 Establece los colores iniciales de pantalla: Papel azul oscuro, tinta cyan brillante.

120-230 Ciclo WHILE...WEND, que al tener en la línea WHILE la condición NOT true, se repite indefinidamente.

Esta repetición, se debe a que al no haber asignado valor alguno a la variable true, ésta adquiere por defecto el valor 0, produciendo la cláusula NOT que true nunca tenga el valor 0, provocando así la repetición indefinida.

150 Chequea el teclado en busca de la pulsación de cualquier botón.

Al ser pulsada una tecla, su código se almacena en la variable alfanumérica a$.

160 Hace que la letra almacenada en a$, sea mayúscula o minúscula, se transforme en minúscula.

Esta línea se introduce, para que en caso de que hallamos pulsado CAPS LOCK, el programa reconozca la tecla almacenada en a$.

180 Consulta si a$ es igual a p, en cuyo caso incrementa en una ciudad la variable p, si el valor de p es mayor que 26, le asigna el valor 0.

190 Efectúa la misma operación que la línea anterior, con la tecla i, y la variable i.

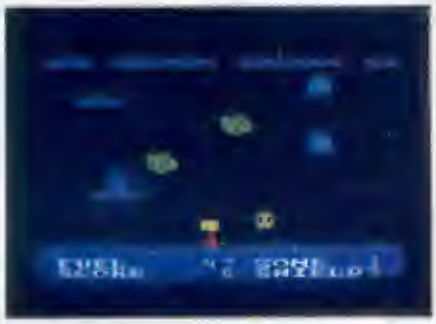
210 INK 0,p cambia el color del papel al número de color p.

INK 1,i cambia el color de la tinta al número de color i.

BORDER p pone el borde de color p.

220 Imprime en la parte superior de la pantalla, el número de color utilizado para la tinta y el usado para el papel.

AMSTER



Amster es un juego estilo Arcade realizado parte en Basic y parte en máquina que nos plantea el conocido desafío de atraversar velozmente una conflictiva zona del espacio interestelar atestada de obstáculos de los más diversos tipos.

Para evitarlos, tenemos a nuestra disposición cañones láser y podemos desplazarnos a derecha e izquierda, además de un oportuno campo de fuerza que impide que los objetos nos destruyan al tocarlos.

— Programa: —Amster I— FJH Software.

— El programa Amster I es una versión del juego Asteroides adaptada al Amstrad CPC-464 y 664, está realizado en Basic y en C.M. y aprovecha lo potencio en el manejo de ventanas del Amstrad.

— El juego se desarrolla a través de 20 zonas en las que progresivamente va aumentando la dificultad.

— La pantalla de juego queda dividida en 2 partes: En una se desarrolla la acción del juego y en la otra se informa al jugador de:

1 Combustible *(que parte de 50).*
2 Zona *(que parte de 0).*
3 Escudo *(que parte de 9).*
4 Puntuación *(que parte de 0).*

— Las teclas para el control de la nave son:
Z: Izquierda.
X: Derecha.
\: Disparo.
ENTER: Escudo protectar.

— También se puede utilizar el Joystick y la tecla ENTER.

— A lo largo del juego irán apareciendo diversos objetos:

1 Asteroide: Aumenta la puntuación en 75.
2 Depósitos de fuel: Aumenta el combustible en 8.
3 Depósita de S. energía: Aumenta la puntuación en 100.
4 Depósito de energía de escudo: Aumenta los escudos en 1.
5 Nubes cósmicas: San pasivas al disparo.
6 Mina cósmica: Al ser alcanzada la onda espansiva acabará con la nave.

Notas:

— Las caracteres de contral se han introducida en el listado pulsando CTRL+letra para ahorrar trabajo a lo hara de teclear y memoria.

— Los caracteres de cantral utilizados han sido:

X CTRL+X
← CTRL+H
↓ CTRL+J

Después de ejecutar el programa, aparecerán en el listada tados los números redefinidos.

PARTES DEL PROGRAMA

NOMBRE	LINEA	FUNCION
Inicialización	80-150	Define ENs y define los objetos
Pantalla	160-200	Configura la pantalla de juego
Controles	210-260	Comprueba el teclado
Controles de vuelo	270-360	Compruebo posible choques
Aporición de objeto	370- 590	Pone aleotoriamente los objetos
Datos de vuelo	600-650	Actualiza el cuadro de mondos
Subrutinas en C.M.	660-800	«Pokea» datos a partir de 30.000
Definición de caracteres	810-1000	Define los objetos con Symbol
Definición de graficos	1010-1140	Forma los objetos y los núms.
Baja fuel	1150-1160	Disminuye la cantidad de fuel
Avanza zona	1170-1200	Pasa a la siguiente zona
Fin de partida	1210-1350	Informa sobre el choque
Disparo	1360-1520	Disparo y comprobaciones
Quitar borrera	1530-1550	Quita borrera después de 3 seg.
Portada	1560-1840	Pone instantáneamente la portada
Reglas	1850-1910	Informa de las teclas y el juego
Explosión chica	1920-1980	Realiza una explosión en los objetos de 2 carocteres
Explosión grande	1990-2130	Realiza una explosión de 4 caracteres.
CLG creciente	2140-2180	Borra la pantalla con un bucle creciente y ORIGIN
Variables	2190-2200	Da valores a las variables del juego

Serie Oro

```
10 '              ****************
20 '              *   AMSTER I   *
30 '              * FJH software *
40 '              ****************
50 '
60 '
70 '
80 SYMBOL AFTER 48
90 GOSUB 670
100 GOSUB 820
110 GOSUB 1020
120 GOSUB 1560
130 GOSUB 2190
140 ENV 1,15,-1,20:ENV 2,10,-1,10:E
NT 1,50,1,1,200,1,1:ENT -2,4,2,1,8,
-2,1,4,2,1
150 CLG
160 '                PANTALLA
170 WINDOW #1,1,20,1,22:PAPER #1,0:
CLS #1:WINDOW #2,1,20,23,25:PAPER #
2,3:CLS #2:BORDER 0
180 PLOT 0,46,10:DRAW 640,46:PLOT 0
,42:DRAW 640,42
190 PEN#2,12:LOCATE#2,2,2:PRINT#2,"
FUEL"SPC(7)"ZONE";:LOCATE#2,2,3:PEN
#2,12:PRINT#2,"SCORE"SPC(6)"SHIELD"
;
200 LOCATE#1,xn,21:PEN#1,2:PRINT#1,
a$;
210 '              CONTROLES
220 IF fe=1 THEN 270
230 IF INKEY(71)=0 OR JOY(0)=4 THEN
 IF xn<3 THEN 300 ELSE LOCATE #1,xn
,22:PRINT#1," ";:LOCATE#1,xn,21:PRI
NT#1," ";:xn=xn-1:GOTO 260
240 IF INKEY(63)=0 OR JOY(0)=8 THEN
 IF xn>18 THEN 300 ELSE LOCATE#1,xn
,22:PRINT#1," ";:LOCATE #1,xn,21:PR
INT#1," ";:xn=xn+1:GOTO 260
250 IF fe=1 OR esc=0 THEN 260 ELSE
IF INKEY(18)=0 THEN fe=1:SOUND 4,10
0,600,3,0,2:AFTER 300,2 GOSUB 1540:
GOTO 300
260 IF fd=1 OR JOY(0)=32 THEN 300 E
LSE IF INKEY(22)=0 THEN fd=1:yd=20:
SOUND 1,30,90,15,2,1:GOSUB 1370
270 '          CONTROL DE VUELO
280 lap=lap+1:IF lap MOD 7=0 THEN G
OSUB 1160
290 IF lap MOD 40=0 THEN GOSUB 1180
300 LOCATE#1,xn,22:PRINT#1," ";:LOC
ATE#1,xn,21:PRINT#1," ";
310 LOCATE #1,1,1:PRINT#1,CHR$(11)
320 LOCATE#1,xn,21:PEN#1,2:PRINT#1,
USING"&";a$;:IF fe=1 THEN PEN#1,14:
LOCATE#1,xn-1,21:SPEED INK 5,5:PRIN
T#1,USING"&";bc$;
330 IF fd=1 THEN GOSUB 1370
340 LOCATE xn,20:CALL 30000:choque=
PEEK(29999)
350 IF choque<>32 AND choque<>232 T
HEN GOSUB 1210
360 IF fe=1 THEN LOCATE xn,19:CALL
30000:cb=PEEK(29999):IF cb<>32 THEN
 IF pun=0 THEN LOCATE#1,xn,19:PRINT
#1," ";:GOTO 370 ELSE pun=pun-5:LOC
ATE#1,xn,19:PRINT#1," ";
370 '          APARICION DE OBJETOS
380 IF fd=1 THEN GOSUB 1370
390 IF flag=1 THEN flag=0:GOTO 220:
ELSE flag=1:PRINT CHR$(22)+CHR$(1);
395 PRINT CHR$(22);CHR$(1);
400 RANDOMIZE TIME
410 IF RND>zonm THEN 430
420 LOCATE#1,1,1:PEN#1,6:PRINT#1,US
ING"&";hn$;:GOTO 220
430 SO=0
440 WHILE so<zonn
450 so=so+1
460 IF RND<zonm THEN 530
470 cu=INT(RND*5)+1
480 ON cu GOTO 490,500,510,520,540
490 RANDOMIZE TIME:LOCATE#1,RND*17+
2,1:PEN#1,11:PRINT#1,USING"&";f$;:G
OTO 530
500 RANDOMIZE TIME:LOCATE#1,RND*17+
2,1:PEN#1,4:PRINT#1,USING"&";s$;:GO
TO 530
510 RANDOMIZE TIME:LOCATE#1,RND*17+
2,1:PEN#1,7:PRINT#1,USING"&";e$;:GO
TO 530
520 RANDOMIZE TIME:LOCATE#1,RND*18+
1,1:PEN#1,5:PRINT#1,USING"&";as$;:G
OTO 530
530 IF RND>zonm THEN 440
540 cu1=INT(RND*4)+1
550 ON cu1 GOTO 560,570,580,590
560 RANDOMIZE TIME:LOCATE#1,RND*17+
2,1:PEN#1,3:PRINT#1,USING"&";m$;:GO
TO 440
570 RANDOMIZE TIME:LOCATE#1,RND*17+
1,1:PEN#1,6:PRINT#1,USING"&";n$;:GO
TO 440
580 RANDOMIZE TIME:LOCATE#1,RND*14+
1,1:PEN#1,6:PRINT#1,USING"&";sn$;:G
OTO 440
585 RANDOMIZE TIME:LOCATE#1,RND*18+
1,1:PEN#1,5:PRINT#1,USING"&";as$;:G
OTO 440
590 WEND
600 '              DATOS DE VUELO
610 LOCATE#2,8,3:PAPER#2,3:PEN#2,8:
PRINT#2,USING"####";PUN;
620 LOCATE#2,18,2:PRINT#2,USING"##"
```

```
;ZONA;
630 LOCATE#2,10,2:PRINT#2,USING"##"
;FUEL;
640 LOCATE#2,19,3:PRINT#2,USING"#";
ESC;
650 GOTO 220
660 '        SUBRRUTINAS EN C.M
670 RESTORE 720
680 FOR e=30000 TO 30012
690 READ a
700 POKE e,a
710 NEXT
720 DATA 62,244,50,47,117,205,96,18
7,208,50,47,117,201
730 RESTORE 790
740 FOR e=30040 TO 30058
750 READ a
760 POKE e,a
770 NEXT
780 DATA 33,0,192,76,76,76,76,195,5
,188
790 DATA 33,4,192,76,76,76,195,5,18
8
800 RETURN
810 '     DEFINICION DE CARACTERRES
820 RESTORE 870
830 FOR x=219 TO 255
840 READ a,b,c,d,e,f,g,h
850 SYMBOL x,a,b,c,d,e,f,g,h
860 NEXT
870 DATA 72,86,162,164,88,66,56,0,4
,130,25,41,4,180,170,18,8,177,34,66
,66,42,17,16,136,68,69,4,72,48,17,1
60,32,80,136,84,42,34,20,16,1,64,0,
41,82,69,37,34,128,12,19,20,37,42,1
0,73,0,32,84,161,80,200,40,77,84,17
0,42,82,162,68,36,24
880 DATA 72,76,85,57,20,149,10,4,65
,24,36,68,50,74,85,173,150,169,77,8
2,36,25,128,18
890 DATA 1,1,1,1,2,2,2,2,0,0,0,24,3
6,66,129,129,128,128,128,128,64,64,
64,64,8,8,8,28,28,28,28,28,62,62,62
,62,127,127,99,65,24,60,60,126,98,1
10,239,231,239,239,110,110,126,60,6
0,24,0,60,102,126,255,153,187,153,2
19
900 DATA 153,255,255,126,102,60,0,6
6,36,153,90,255,255,153,187,153,221
,153,255,90,153,36,66,16,168,85,170
,85,42,85,0,16,170,85,170,85,170,81
,0
910 DATA 0,170,84,170,85,170,20,0,2
3,123,127,254,255,247,111,126,192,2
24,224,248,56,220,252,252,126,238,2
54,252,124,248,192,192
920 DATA 127,127,47,61,62,31,15,3,6
6,66,36,36,126,110,239,239,239,239,
102,126,36,36,66,66,240,248,248,252
,252,254,254,255
930 DATA 15,31,31,63,63,127,127,255
,255,247,247,243,243,241,241,240,25
5,239,239,207,207,143,143,15,16,16,
16,16,16,40,0,0
940 RESTORE 990
950 FOR x=48 TO 57
960 READ a,b,c,d,e,f,g,h
970 SYMBOL x,a,b,c,d,e,f,g,h
980 NEXT
990 DATA 60,36,36,36,36,36,60,0,24,
24,8,8,8,8,8,0,60,4,4,60,32,32,60,0
,60,4,4,60,4,4,60,0,36,36,36,60,4,4
,4,0,60,32,32,60,4,4,60,0,60,32,32,
60,36,36,60,0,60,4,4,4,4,4,4,0,60,3
6,36,60,36,36,60,0,60,36,36,60,4,4,
4,0
1000 RETURN
1010 '     DEFINICION DE GRAFICOS
1020 a$=CHR$(234)+"
"+CHR$(235)
1030 f$=CHR$(236)+"
"+CHR$(237)
1040 e$=CHR$(240)+"
"+CHR$(241)
1050 s$=CHR$(238)+"
"+CHR$(239)
1060 n$=CHR$(242)+CHR$(243)+CHR$(24
4)
1070 sn$=CHR$(242)+CHR$(243)+CHR$(2
43)+CHR$(243)+CHR$(243)+CHR$(244)
1080 hn$=n$+" "+sn$+" "+sn$+" "+CHR
$(242)+CHR$(243)
1090 m$=CHR$(249)+"
"+CHR$(250)
1100 as$=CHR$(245)+CHR$(246)+"
"+CHR$(248)+CHR$(247)
1110 bc$=CHR$(22)+CHR$(1)+CHR$(231)
+"
"+CHR$(232)+"
"+CHR$(233)+CHR$(22)+CHR$(0)
1120 ega$=CHR$(225)+CHR$(226)+"
"+CHR$(227)+CHR$(228):ego$=CHR$(229
)+"
"+CHR$(230):ema$=CHR$(224)+CHR$(223
)+"
"+CHR$(222)+CHR$(221):emo$=CHR$(220
)+"
"+CHR$(219)
1130 bl$="  "+"
"+"  "
1140 RETURN
1150 '              BAJA FUEL
1160 IF fuel=0 THEN GOTO 1210 ELSE
fuel=fuel-1:RETURN
1170 '              AVANZA ZONA
1180 zona=zona+1:IF zona=21 THEN 12
50
1190 zonn=zonn+0.1:zonm=zonm+0.02
1200 RETURN
1210 '         FIN DE PARTIDA
```

```
1220 SPEED INK 4,4:PAPER#1,0:PEN#1,
15:LOCATE#1,xn,21:PRINT#1,emo$;
1230 SOUND 132,0,0,0:SOUND 129,0,0,
0
1240 PAPER#1,6:PEN #1,4:FOR e=0 TO
12:SOUND 2,500,20,15,1,0,RND*10+20:
BORDER 6,24:INK 0,24,6:CALL 30050:F
OR T=0 TO 35:NEXT:CALL 30040:NEXT:S
OUND 2,500,250,15,1,0,31:LOCATE#1,6
,13:PRINT#1,"GAME OVER":FOR T=0 TO
4000:NEXT
1250 GOSUB 2150
1260 BORDER 0:INK 0,0
-INFOR
-FIN  D
";:LOCATE 2,6:PEN 11:PR
INT"-CAUSA:":IF zona=21 THEN 1300
1280 LOCATE 2,10:PEN 8:IF fuel=0 TH
EN PRINT"-FALTA DE FUEL" ELSE IF bl
anco=250 THEN PRINT"-DESTRUCCION DE
 MINA" ELSE PRINT"-CHOQUE FRONTAL"
1290 LOCATE 2,15:PEN 11:PRINT"-ZONA
:";:PEN 8:PRINT zona:GOTO 1310
-LLEGA
":L
OCATE 2,21:PEN 4:PRINT"Quieres juga
r de    nuevo?":GOTO 1320
1310 LOCATE 2,21:PEN 4:PRINT"Quiere
s intentarlo  otra vez?"
1320 zz$=INKEY$:zz$=UPPER$(zz$)
1330 IF zz$="S" THEN GOSUB 2150:GOS
UB 2200:GOTO 150
1340 IF zz$="N" THEN GOSUB 2150:END
1350 GOTO 1320
1360 '          DISPARO
1370 IF yd=20 THEN xd=xn
1380 lap=lap+1:IF lap MOD 3=0 THEN
GOSUB 1160
1390 IF lap MOD 40=0 THEN GOSUB 118
0
1400 PAPER#1,0:PEN#1,12:LOCATE#1,xd
,yd:PEN#1,12:PRINT#1,CHR$(255);
1410 LOCATE xd,yd-1:CALL 30000:blan
co=PEEK(29999)
1420 IF blanco=32 THEN 1500
1430 IF blanco=242 OR blanco=243 OR
 blanco=244 OR blanco=227 OR blanco
=228 OR blanco=221 OR blanco=222 OR
 blanco=230 OR blanco=219 THEN SOUN
D 2,500,120,15,1,0,1:LOCATE#1,xd,yd
:PRINT#1," ";:fd=0:RETURN
1440 IF blanco=237 THEN SOUND 2,500
,120,15,1,0,5:LOCATE#1,xd,yd:PRINT#
1," ";:fd=0:GOSUB 1930:IF fuel>=75
THEN RETURN ELSE fuel=fuel+8:RETURN
1450 IF blanco=241 THEN SOUND 2,500
,120,15,1,0,10:LOCATE#1,xd,yd:PRINT
#1," ";:GOSUB 1930:pun=pun+100:fd=0
:RETURN
1460 IF blanco=239 THEN SOUND 2,500
,120,15,1,0,15:LOCATE#1,xd,yd:PRINT
#1," ";:fd=0:GOSUB 1930:IF esc=9 TH
EN RETURN ELSE esc=esc+1:RETURN
1470 IF blanco=250 THEN LOCATE#1,xd
,yd:PRINT#1," ";:GOSUB 1930:GOTO 12
20
1480 IF blanco=248 THEN SOUND 2,500
,120,15,1,0,31:LOCATE#1,xd,yd:PRINT
#1," ";:fd=0:GOSUB 2000:pun=pun+75:
RETURN
1490 IF blanco=247 THEN SOUND 2,500
,120,15,1,0,31:LOCATE#1,xd,yd:PRINT
#1," ";:fd=0:GOSUB 2070:pun=pun+75:
RETURN
1500 LOCATE#1,xd,yd:PAPER#1,0:PRINT
#1," ";:IF yd=2 THEN fd=0:yd=20:RET
URN
1510 yd=yd-0.5
1520 RETURN
1530 '          QUITAR BARRERA
1540 fe=0:PAPER#1,0:LOCATE#1,xn-1,2
1:PRINT#1," ";:LOCATE#1,xn,20:PRINT
#1," ";:LOCATE#1,xn+1,21:PRINT#1,"
";
1550 IF esc=0 THEN RETURN ELSE esc=
esc-1:RETURN
1560 '              PORTADA
1570 SPEED INK 30,30
1580 FOR e=0 TO 15:INK e,0:NEXT:BOR
DER 0:MODE 0:PEN 5:PAPER 0
1590 RESTORE 1660
1600 FOR n=1 TO 5
1610 READ pin:PEN pin
1620 FOR m=1 TO 18
1630 READ a:LOCATE m,n:PRINT CHR$(a
);
```

PRINCIPALES VARIABLES

VARIABLES	FUNCION
XN	Caardenada x de la nave
PUN	Puntuación
ZONA	Zona en la que se encuentra la nave
ZONM	Coeficiente de probabilidad de aparición de los objetos
ZONN	N.º de objetos que aparecen en cada fila
FD	Flag de disparo activodo
FE	Flag de escudo activado
Fuel	Combustible de la nave
ESC	N.º de escudos disponibles
LAP	Controlador de disminución de fuel y aumenta de zona *(por MOD)*

```
1640 NEXT
1650 NEXT
1660 DATA 1,143,143,32,133,138,32,1
43,143,32,143,143,133,143,143,32,14
3,143,32,2,133,138,32,251,252,32,14
3,32,32,32,133,32,143,32,32,133,138
,32,4
1670 DATA 143,143,32,253,254,32,143
,143,32,32,133,32,143,143,32,143,14
3,32,8,133,138,32,133,138,32,32,138
,32,32,133,32,143,32,32,133,133,32,
12
1680 PEN 6:LOCATE 19,1:PRINT CHR$(1
43);CHR$(143);:LOCATE 19,2:PRINT CH
R$(209);CHR$(211);:LOCATE 19,3:PRIN
T CHR$(209);CHR$(211);:LOCATE 19,4:
PRINT CHR$(209);CHR$(211);:LOCATE 1
9,5:PRINT CHR$(143);CHR$(143);
1690 DATA 133,138,32,133,138,32,143
,143,32,32,133,32,143,143,32,133,13
8,32,143,143
1700 RESTORE 1750:PEN 8
1710 FOR e=1 TO 9
1720 READ x,y,c$
1730 LOCATE x,y:PRINT c$;
1740 NEXT
1750 DATA 4,7,"TU NAVE  MINA",4,10,
NUBE,4,11,COSMICA,4,13,FUEL,4,16,ES
CUDO,4,19,SUPER,4,20,ENERGIA,4,22,A
STEROIDE,13,8,COSMICA
1760 PEN 13:LOCATE 2,25:PRINT"R-Reg
las  J-Juego"
1770 PEN 2:LOCATE 3,7:PRINT a$:PEN
6:LOCATE 1,10:PRINT n$:PEN 11:LOCAT
E 3,13:PRINT f$:PEN 7:LOCATE 3,16:P
RINT e$:PEN 4:LOCATE 3,19:PRINT s$:
PEN 5:LOCATE 2,22:PRINT as$:PEN 10:
LOCATE 12,7:PRINT m$
1780 RESTORE 1800
1790 FOR e=1 TO 12:READ col:INK e,c
ol:NEXT
1800 DATA 3,6,1,15,12,13,14,24,0,10
,2,26
1810 INK 13,1,15:INK 14,24,6:INK 15
,25,3
1820 IF INKEY(50)=0 THEN 1860
1830 IF INKEY(45)=0 THEN GOSUB 2150
:GOTO 130
1840 GOTO 1820
1850 '                REGLAS
1860 GOSUB 2150:MODE 1:WINDOW 5,36,
1,25:INK 1,24:INK 2,18:INK 3,26:PEN
 1
":PE
N 2
1880 LOCATE 1,5:PRINT"En tu viaje h
acia la ZONA 20,tussistemas de nave
gacion espacial han dejado de funci
onar y te hasintroducido en un sect
or de altopeligro,ahora deberas luc
har contra insospechados peligros e
n elSECTOR DIABOLICO."
1890 LOCATE 1,15:PEN 1:PRINT"CONTRO
"SP
C(6)"-IZQUIERDA":LOCATE 11,19:PRINT
"SPC(6)"-DERECHA":LOCATE 11,21:PR
"SPC(6)"-FUEGO":LOCATE 11,23:P
  -ESCUDO"
1900 LOCATE 7,25:PEN 2:PRINT"[ PULS
A     SPACE ]";
1910 zz$=INKEY$:IF zz$=" " THEN INK
 1,0:GOSUB 2150:GOTO 1580 ELSE 1910
1920 '          EXPLOSION CHICA
1930 SPEED INK 5,5:RANDOMIZE TIME:I
F RND>0.5 THEN 1960
1940 LOCATE#1,xd,yd-2:PEN#1,INT(RND
*2)+14:PRINT#1,USING"&";emo$;
1950 GOTO 1970
1960 LOCATE#1,xd,yd-2:PEN#1,INT(RND
*2)+14:PRINT#1,USING"&";ego$;
1970 FOR e=0 TO 80:NEXT:LOCATE#1,xd
,yd-1:PRINT#1," ";:LOCATE#1,xd,yd-2
:PRINT#1," ";
1980 RETURN
1990 '         EXPLOSION GRANDE
2000 SPEED INK 5,5:RANDOMIZE TIME
2010 LOCATE#1,xd,yd-2:PEN#1,INT(RND
*2)+14:IF RND>0.5 THEN 2040
2020 PRINT#1,USING"&";ema$;
2030 GOTO 2050
2040 PRINT#1,USING"&";ega$;
2050 FOR e=0 TO 80:NEXT:LOCATE#1,xd
,yd-2:PRINT#1,USING"&";bl$;
2060 RETURN
2070 SPEED INK 5,5:RANDOMIZE TIME
2080 LOCATE#1,xd-1,yd-2:PEN#1,INT(R
ND*2)+14:IF RND>0.5 THEN 2110
2090 PRINT#1,USING"&";ema$;
2100 GOTO 2120
2110 PRINT#1,USING"&";ega$;
2120 FOR e=0 TO 80:NEXT:LOCATE#1,xd
-1,yd-2:PRINT#1,USING"&";bl$;
2130 RETURN
2140 '           CLG CRECIENTE
2150 FOR e=100 TO 338 STEP 16:s=s+1
4:ORIGIN 0,0,320+e,320-e,200+s,200-
s:CLG 9:NEXT
2160 BORDER 0
2170 ORIGIN 0,0,0,640,0,400:s=0
2180 RETURN
2190 '            VARIABLES
2200 xn=10:pun=0:zona=1:zonn=1:zonm
=0.1:fd=0:fe=0:flag=1:fuel=50:esc=9
:s=0:lap=2
2210 RETURN
```

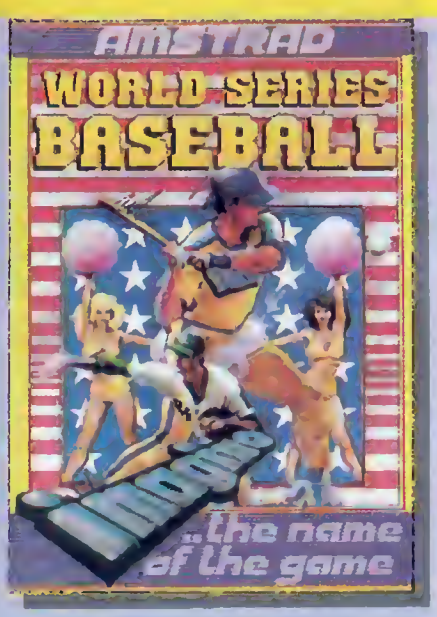
¡NUEVO!

SIEMPRE LOS PRIMEROS EN TENER LO ULTIMO

circulo de soft

MICROAMIGO S.A.

P.° de la Castellana, 268, 3.° C. 28046-MADRID.
Tel.: (91) 733 25 00



BASEBALL

Impresionante simulación en 3 dimensiones. Se puede competir contra el ordenador o contra otro jugador.
No es necesario conocer el beisbol.
Hay un modo de demostración.
Pantallas gigantes para ver de cerca la acción.

P.V.P.: 2.100 ptas.
Precio C. de Soft: 1.890 ptas.

RAID

Defiéndete con tu escuadrilla de aviones del ataque nuclear que han lanzado sobre ti. Tu viaje estará lleno de peligros hasta que llegues a las bases de lanzamiento de misiles enemigas. Tendrás que destruirlos para salvar a tu país de una catástrofe nuclear. Gráficos y acción sensacionales.

P.V.P.: 2.300 ptas.
Precio C. de Soft: 2.070 ptas.

MATCH DAY

¡Ahora para Amstrad! No se trata de un juego de fútbol cualquiera. Fantástica acción en 3 dimensiones y animación total que dan vida al fútbol. Quedarás maravillado con el control del balón y desarrollarás tu destreza y técnica jugando contra otro jugador o contra el ordenador.

P.V.P.: 2.300 ptas.
Precio C. de Soft: 2.070 ptas.

¡¡¡...Y LOS TRES PROGRAMAS POR SOLO 5.400 PTAS!!!

¡HAZTE HOY MISMO SOCIO DEL CIRCULO DE SOFT! Además de poder adquirir tus programas al mejor precio, recibirás información de forma periódica y gratuita, del mejor software que aparezca en el mercado.

¿QUE HAY QUE HACER PARA SER SOCIO DEL CIRCULO DE SOFT? Así de fácil: envíanos por correo tu nombre, dirección y modelo de ordenador, o bien, pide por teléfono o por correo tu primer programa. ¡Y entrarás a formar parte del CIRCULO.DE SOFT de forma inmediata!

☐ Sí, quiero ser SOCIO desde hoy mismo del CIRCULO DE SOFT y recibir periódicamente información de novedades de software, así como beneficiarme desde hoy mismo de los precios reducidos reservados a los SOCIOS y de sus Ofertas Especiales. El ser SOCIO no me obliga a compra alguna.

Si prefieres formalizar tu compra por teléfono puedes hacerlo llamando al (91) 733 25 00. ¡¡NO SE COBRAN LOS GASTOS DE ENVIO POR CORREO!!

TITULO	P.V.P.	ORDENADOR

☐ Contrarreembolso ☐ Giro Postal ☐ Talón adjunto a Microamigo, S.A. ☐ Tarjeta VISA n.º _____ Fecha caducidad _____

Nombre _____ Apellidos _____ Edad _____

Domicilio _____ Teléfono _____

Localidad _____ C.P. _____ Provincia _____

EL STACK

Después de un largo descanso en el estudio del microprocesador Z80 y su lenguaje aplicado a los Amstrads, durante el cual se han visto desde rutinas muy sofisticadas de movimiento de gráficos y sprites, hasta detallados análisis de las mejores herramientas para el programador que existen en nuestro mercado, ya va siendo hora de retomar el hilo de nuestra narración, ponerse serios delante del teclado armados con un ensamblador y aprender un poco más.

n este artículo nos vamos a plantear el estudio de una parte muy especial del Z80, cuyo conocimiento y dominio es esencial para el programador en lenguaje máquina: el **STACK**.

Esta palabra tan rara, que en español se suele traducir como **pila** o **pila de máquina,** referencia y describe una estructura de datos conocida desde hace mucho en informática y que los anglosajones, tan aficionados como son a las obreviaturas, denominan estructura LIFO, del inglés **Last In, First Out.**.

Esto significa que los datos se organizan en la memoria de tal manera que el último en ocupar un puesto en la zona del stack es el primero en ser recuperado.

¿Qué es una pila?

Normalmente la lectura de la frasecita anterior, así a primera vista, no resulta muy clara que digamos, por lo que voy a recurrir a un ejemplo que he visto reproducido en el 90 por 100 de los libros que tratan de código máquina: la pila de platos.

Imaginemos que tenemos sobre una mesa una caja de madera, en la que caben justo 5 platos puestos uno encima de otro.

La caja representa un stack con espacio para 5 elementos y los platos, obviamente, son los números que colocaremos en el stack.

Sigamos imaginando: metemos dos platos en la caja, dos números en el stack.

Mucho más obviamente que antes, el segundo plato quedará encima del primero, por lo que si sacamos un plato de la caja obtendremos el segundo de ellos, es decir, el último que se metió en la caja. *(Ultimo en entrar, primero en salir.)*

Para obtener el primer plato, el que está en el fondo de la caja, deberemos extraer antes el segundo, que está encima. Por esta razón, si uno inspecciona un stack, siempre se encuentra a mano con el último dato metido en él y sacar otros más **«profundos»** requiere normalmente echar del stack todos los números que están **encima.**

Bien. Una vez comprendido intuitivamente *(espero)* lo que es un stack, surgen dos preguntas muy lógicas:

1. ¿Para qué demonios sirve una estructura de datos tan esotérica?

2. ¿Cuál es la representación de la caja de platos en la memoria de un ordenador?

Trataremos de contestar en primer lugar a la segunda pregunta y un poco más tarde, a la primera.

Crecer hacia abajo

Las estructuras LIFO normalmente se implementan en la memoria haciéndolas crecer hacia abajo. Es decir, se coloca un puntero al principio de la zona de memoria donde se crea el stack, el llamado STACK POINTER *(registro SP del Z80);* cuando se introduce un número en el stack, el puntero se decrementa, de modo que siempre apunta al último introducido. Cuando se extrae un número del stack, el SP se incrementa apuntando de nuevo al siguiente elemento de la pila.

F. L. Frontán

Código máquina

La ventaja de este sistema estribo en el hecho de poder separar lo más posible la zona de programa de la del stack, para evitar que uno se **coma** al otro, con resultados fatales.

En Basic, el sistema operativo se encarga de controlar que esto no ocurra, pero en máquina es el programador el que lleva la batuta y debe cuidarse él mismo.

Las instrucciones del Z80 que afectan al stack pueden dividirse en dos grupos:

1. Instrucciones que alteran el número de elementos existentes en el stack.

2. Instrucciones que lo **manipulan,** , sin que necesariamente alteren el **grosor** de la pila.

A las primeras pertenecen las siguientes órdenes:

PUSH, POP, CALL y RET

La instrucción PUSH sirve para introducir un número en el stack. En el caso del Z80, esta orden se aplica siempre a los registros pares:

PUSH AF PUSH DE PUSH IX
PUSH BC PUSH HL PUSH IY

y el efecto que se consigue es que el puntero de pilo *(registro SP)* se decremente en dos unidades después de que el contenido del registro ho sido colocado en el stack.

Por ejemplo:

HL contiene &A0B0: LD HL, &A000

SP apunta a &0400: SP‖&0400

Contenido de &0400=0: (&0400)=0

Orden PUSH HL: (&03FF)=&A) (&03FE)=&B0

SP apunta ahora a &03FE: SP‖&03FE

En definitiva, como comentamos antes, el número se coloca en el stack o pila y el registro SP se decrementa en dos unidades.

Respecto a la instrucción POP, realiza el efecto inverso a la instrucción PUSH. Sólo puede aplicarse a registros dobles y si miramos el ejemplo anterior de **«abajo arriba»**, estaremos observando lo que ocurre en el stack cuando se ejecuta una instrucción POP. El número extraído del stack se carga en el registro **popeado,** y el SP se incremento en dos unidades.

Así, yo puedo corgar cualquier registro con cualquier otro número o registro de una manera muy simple; por ejemplo:

PUSH IX
POP HL

conseguiró el mismo efecto que la inexistente instrucción:

LD HL, IX

Las órdenes CALL y RET tienen entre sí el mismo «efecto espejo» que PUSH y POP.

La primera, CALL, sirve poro llamar a una subrutino, mientras que la segundo RET, transmite el flujo de los datos al programa principal.

Supongamos que en lo posición de memoria &A000 introducimos el mnemónico: CALL BB5A.

Esta instrucción ocupa 3 bytes, por lo que el panorama de la memorio será el siguiente:

DIRECCION	OPCODE
&A000	&CD
&A001	&5A
&A002	&BB
&A003	Resto del programa

Cuando el Z80 llega o &A000, almaceno en el stack lo dirección &A003, que es donde hay que volver cuando la subrutino termine su labor, es decir, dos bytes más allá del opcode &CD; como si hubiéramos hecho un PUSH, vaya. El SP se decrementa en dos unidades *(crece)*.

A estas alturas, el Z80 está ejecutando la rutina que comienza en BB5A, hasta que se encuentra con lo instrucción RET. Cuando sucede esto, se extrae del stack el número almacenado en la cima del mismo, esto es, &A003, el SP se incrementa dos posiciones y el programa regresa o RESTO DEL PROGRAMA. El efecto en el stack es muy parecido a la instrucción POP.

La orden RET podríamos llamarla retorno incondicional. Como en el caso de las órdenes de salto, sean relativos o absolutos, existen retornos condicionales, los mismo que en aquéllos.

Los mnemónicos son tombién muy parecidos, por ejemplo:

RET C

significa retorno si el Corry está activado. Lo misma situación se observa en las instrucciones CALL, también existen llamadas condicionales:

CALL NZ

significa **bifurca y retorna** si el indicador de cero está a cero.

El resto de las instrucciones que afectan de algún modo al stack vamos a verlos en los programas, ejemplo acto seguido, aprovechando de paso para explicar algunas circunstancias en la que el stack puede ser útil.

Es el más sencillo de todos y muestra cómo llamar indirectamente a una subrutina y alterar así el flujo del programa de forma **«anormal»**. Para mayor claridad, consta de dos partes, ensambladas en zonas muy diferentes de la memoria.

La rutina que comienza en A000 simplemente emite un sonido por el altavoz, usando la vieja conocida del firmware BB5A.

La primera parte del programa, ensamblada en la dirección 8000, es la que realiza el **truquito.**

Lo que hacemos es cargar un registro cualquiera, en este caso DE, con la dirección de comienzo de la rutina a la que queremos llamar indirectamente (A000) y luego metemos esa dirección en el stack, alterando el puntero de pila *(SP)*.

Cuando el Z80 llega al RET, hace lo que anteriormente vimos: extrae el dato de 16 bits de la cima de la pila y **«salta»** allí, donde continuará ejecutando lo que haya.

Como casualmente el primer dato de la pila es A000, pues el programa salta allí, pero sin **«enterarse»** directamente de que se ha llamado a una subrutina independiente.

En la última orden de ésta, JP BB5A, no es necesario poner CALL en lugar de JP, porque BB5A posee su propio RET.

El programa dos es una copia casi exacta del anterior, pero con dos diferencias importantes: el uso de la orden EX (SP), HL y que... no funciona.

La instrucción EX (SP), HL realiza un intercambio del dato de 16 bits contenido en las dos posiciones de memoria a partir de la que apunte SP con el contenido de HL.

Uno podría pensar que, en principio, valdría para hacer la tarea del programa 1, pero obsérvese que el stack ni crece ni decrece con la instrucción de intercambio.

Por tanto, si ejecutamos este programa desde Basic con CALL &8000, la orden de intercambio se **carga** la dirección de retorno puesta en el stack por el propio comando Basic CALL. Resultado: cuelgue.

Más claro no se puede ver la diferencia entre las órdenes de intercambio con el stack y las PUSH.

El programa 3 sí que funciona y muestra una técnica muy usual que, empleando EX (SP), HL permite usar el stack como almacén temporal de datos, que posteriormente serán recogidos por otra rutina.

La rutina STACK llama primero a una del firmware que espera a que pulsemos una tecla y la almacena en el acumulador.

Una vez hecho esto, cargamos HL con el valor ASCII de la tecla en el orden correcto *(atención: A va al registro H, no al L)*, y ahora viene la **astucia:** primero, metemos en el stack el valor de HL, o sea, la tecla pulsada y, luego, pusheamos HL para volver a meter en él la dirección correcta de retorno.

A final de esta operación, en la cima de la pila está la dirección de retorno e inmediatamente **debajo,** el valor de la tecla pulsada.

Así, tras la llamada CALL STACK en la rutina principal, la orden POP AF deja el valor de la tecla en el acumulador. Sólo basta imprimirla en pantalla para comprobarlo.

Este ejemplo concreto es bastante trivial, pero la técnica es muy útil, y tarde o temprano os encontraréis con la necesidad de usarla en vuestros programas.

Este programita también es muy sencillo, pero muestra un uso del stack bastante raro.

Recordemos: el stack no es más que una zona de la memoria como cualquier otra, sólo que organizada de una manera especial. Por tanto, **¿qué pasa si yo coloco la dirección de comienzo de dicha área en la pantalla y luego hago un PUSH repetidamente?** Pues pasa que lleno la pantalla con el número contenido en HL, pero lo hago de dos bytes en dos bytes en lugar de byte a byte. Es decir, estoy volcando un gráfico en la pantalla *(¿a que os suena a programas de juegos?)*

Para ello hemos usado tres órdenes que involucran al stack:

a) LD (STACK), SP. Preserva el valor del puntero de pila en una variable para poder recuperarlo luego. ¡Que el stack estuviera siempre en la pantalla no sería una buena estrategia de programación que digamos!

b) LD SP,O. Esta instrucción permite inicializar la dirección a la que apunta SP, a cualquier parte del espacio de 64 K del **Amstrad**. Lo colocamos en la cero porque la orden PUSH HL que viene después hace que SP se decremente, y si decrementamos cero nos sale &FFFF (si, la zona de la pantalla).

c) LD SP, (STACK). Restituye al puntero de pila su valor original.

«HACKER»

Persona que se introduce o interfiere ilegalmente en redes de ordenadores con ánimo de lucro.

PROGRAMAS

Aquí mostramos cómo implementar un menú de opciones basado en la técnica vista en el programa I.

El método es sencillo: primero hacemos como en el programa I. Metemos en el stack la dirección de la rutina a la que queremos que se **«retorne»** siempre, en este caso OPCION.

La rutina TECLA espera a que se pulse una teclo comprendida entre el 0 y el 9 y retorna... a OPCION, la cual nos informa de que hemos elegido la opción tal del menú y acaba.

Esto basta para mostrar la técnica, pero en una aplicación real, en lugar del RET del de la rutina OPCION debiera haber una instrucción JP de bifurcación a la rutina elegida en el menú.

No la hemos incluido porque la forma de implementar esto involucra estructuras de datos como tablas de salto y listas encadenadas que, por sí mismas, constituyen material más que sobrado para varios artículos.

De cualquier forma, como ejercicio, podría tratar de imaginar cómo nos las compondríamos para lograrlo. **¿Tal vez almacenando en el stack todas las direcciones de salto a las nuevas rutinas como en el programa III?**

Esto es una miniherramienta que nos dirá en cualquier momento la dirección de memoria a la que apunta el SP y lo que contiene. No la comentamos más ampliamente porque los lectores de nuestro curso de código máquina deben saber ya cómo funciona. Además, la rutina **«hexa»** ya apareció en otro artículo. Bueno, un ejercicio más.

Por cierto, **¿alguien se atrevería a rizar el rizo y meter esta rutina en un nueva comando Basic mediante el RSX?**

El programa VII no tiene desperdicio. Hace un uso intensísimo del stack, resumiendo en gran parte lo que se ha visto en los anteriores, pero esboza también algo muy importante: cómo implementar la RECURSIVIDAD en lenguaje máquina *(ojo, hay más maneras de hacerlo)*, la rutina MAS.

Proponemos a los lectores un nuevo ejercicio: traten de comprender cómo y por qué esta rutina funciona. Si lo consiguen, el concepto del stack nunca más tendrá secretos para ellos. Un consejo: ejecútese el programa VII antes de intentar comprenderlo, y, sobre todo, recuérdese lo que sucede cuando el Z80 se encuentra con la instrucción RET.

```
8000             10          ORG  #8000
8000             20          ENT  #8000
8000  1100A0      30          LD   DE,#A000
8003  D5          40          PUSH DE
8004  C9          50          RET
A000              60          ORG  #A000
A000  3E07        70          LD   A,7
A002  C35ABB      80          JP   #BB5A
```

```
8000             10          ORG  #8000
8000             20          ENT  #8000
8000  2100A0      30          LD   HL,#A000
8003  E3          40          EX   (SP),HL
8004  C9          50          RET
A000              60          ORG  #A000
A000  3E07        70          LD   A,7
A002  C35ABB      80          JP   #BB5A
```

```
8000             10          ORG  #8000
8000             20          ENT  #8000
8000  CD0B80      30          CALL STACK
8003  F1          40          POP  AF
8004  CD5ABB      50          CALL PRINT
8007  C9          60          RET
8008  CD06BB      70 STACK:   CALL ESPERA
800B  67          80          LD   H,A
800C  2E00        90          LD   L,0
800E  E3          100         EX   (SP),HL
800F  E5          110         PUSH HL
8010  C9          120         RET
BB5A              130 PRINT:  EQU  #BB5A
BB06              140 ESPERA: EQU  #BB06
```

```
8000             10          ORG  #8000'
8000             20          ENT  #8000
8000  21FFFF      30          LD   HL,#FFFF
8003  ED731480    40          LD   (STACK),SP
8007  310000      50          LD   SP,0
800A  0664        60          LD   B,100
800C  E5          70 MAS:     PUSH HL
800D  10FD        80          DJNZ MAS
800F  ED7B1480    90          LD   SP,(STACK)
8013  C9          100         RET
8014  0000        110 STACK:  DEFW 0
8016              120         END
```

```
8000             10          ORG  #8000
8000             20          ENT  #8000
8000  2110B0      30          LD   HL,OPCION
8003  E5          40          PUSH HL
8004  CD06BB      50 TECLA:   CALL ESPERA
8007  FE30        60          CP   "0"
8009  38F9        70          JR   C,TECLA
800B  FE3A        80          CP   "9"+1
800D  30F5        90          JR   NC,TECLA
800F  C9          100         RET
8010  212680      110 OPCION: LD   HL,MENS
8013  E5          120         PUSH AF
8014  CD1C80      130         CALL NEXT
8017  F1          140         POP  AF
8018  CD5ABB      150         CALL PRINT
801B  C9          160         RET
801C  7E          170 NEXT:   LD   A,(HL)
801D  FEFF        180         CP   #FF
```

```
801F  C8          190         RET  Z
8020  CD5ABB      200         CALL PRINT
8023  23          210         INC  HL
8024  18F6        220         JR   NEXT
8026  484153204F 230 MENS:    DEFM "HAS
                                SELECCIONADO LA OPCION= "
8042  FF          240         DEFB #FF
BB06              250 ESPERA: EQU  #BB06
BB5A              260 PRINT:  EQU  #BB5A
8043              270         END
```

```
8000             10          org  #8000
8000             20          ent  #8000
8000  210C80      30          ld   hl,mens1
8003  CD4980      40          call next
8006  ED73B780    50          ld   (stack),sp
800A  2AB780      60          ld   hl,(stack)
800D  67          70          ld   h,a
800E  CD7480      80          call hexa
8011  6F          90          ld   l,a
8012  CD7480      100         call hexa
8015  3E0A        110         ld   a,10
8017  CD5ABB      120         call print
801A  3E0D        130         ld   a,13
801C  CD5ABB      140         call print
801F  216B80      150         ld   hl,mens2
8022  CD4980      160         call next
8025  2AB7B0      170         ld   hl,(stack)
8028  5E          180         ld   e,(hl)
8029  2B          190         dec  hl
802A  56          200         ld   d,(hl)
802B  7A          210         ld   a,d
802C  CD7480      220         call hexa
802F  7B          230         ld   a,e
8030  CD7480      240         call hexa
8033  C9          250         ret
8034  F5          260 hexa:   push af
8035  0F          270         rrca
8036  0F          280         rrca
8037  0F          290         rrca
8038  0F          300         rrca
8039  CD3D80      310         call nibble
803C  F1          320         pop  af
803D  E60F        330 nibble: and  #0f
803F  C690        340         add  a,#90
8041  27          350         daa
8042  CE40        360         adc  a,#40
8044  27          370         daa
8045  CD5ABB      380         call print
8048  C9          390         ret
8049  7F          400 next:   ld   a,(hl)
804A  FEFF        410         cp   #ff
804C  C8          420         ret  z
804D  CD5ABB      430         call print
8050  23          440         inc  hl
8051  18F4        450         jr   next
BB5A              460 print:  equ  #bb5a
805?  ?0?04F54    470 mens1:  defm "PUNTERO DE
                                PILA= &"
8065  FF          480         defb #ff
80?5  ?4?4F4?20   490 mens2:  defm "CONTENIDO
                                DEL PUNTERO DE PILA= &"
8086  FF          500         defb #ff
80B7  0000        510 stack:  defw 0
80B9              520         end
```

```
8000             10          ORG  #8000
8000             20          ENT  #8000
8000  211F80      30          LD   HL,TEXT
8003  ED7323B1    40          LD   (STACK),SP
8007  3122B1      50          LD   SP,BUF1
800A  CD1280      60          CALL MAS
800D  ED7B23B1    70          LD   SP,(STACK)
8011  C9          80          RET
8012  E5          90 MAS:     PUSH HL
8013  23          100         INC  HL
8014  7E          110         LD   A,(HL)
8015  FEFF        120         CP   #FF
8017  C412B0      130         CALL NZ,MAS
801A  E1          140         POP  HL
801B  7E          150         LD   A,(HL)
801C  C35ABB      160         JP   #BB5A
801F  484F4C41    170 TEXT:   DEFM "HOLA"
8023  FF          180         DEFB #FF
8024              190 BUF:    DEFS 254
8122              200 BUF1:   DEFS 1
8123  0000        210 STACK:  DEFW 0
```

Sin duda alguna

A través de esta sección se pretende resolver, en la medida de lo posible, todas las posibles dudas que **«atormenten»** a todas las personas interesadas en el mundo del AMSTRAD, sean o no poseedores de uno y, si lo son, se encuentren en cualquier nivel de destreza en su manejo.

Semanalmente, aparecen en estas páginas las consultas de la mayor cantidad de usuarios posible; ello redundará en un mejor servicio y en un contacto más estrecho entre todos nosotros a través de la revista.

SIN DUDA ALGUNA está abierta a todos.

En primer lugar quiero felicitaros por esta magnífica revista de la cual sois padres y de la cual voy a ser un gran asiduo. «Hace poco me compré un CPC/664 y mientras no aprendo a manejarlo bien me gustaría utilizarlo para jugar pero tengo ciertas dudas sobre los programas de juegos:

1. ¿Existen muchos programas de juegos para el 664?
2. ¿Son más caros que las cintas? ¿Sobre qué precio?
3. ¿Se pueden pasar los programas en cinta para disco?»

Gracias anticipadas.

Un amigo:

José Antonio

1. Depende de lo que entiendas por muchos. Entre juegos y gestión, más de 100 programas disponibles en las tiendas. Indescomp podrá darte una información mucho más precisa de este tema.

Su dirección es:

Avd. Mediterráneo, n.º 9. Madrid. Tel.: (91) 433 44 58

2. Los programas en disco siempre son más caros que los programas en cinta, ya que el soporte, el disco, cuesta mucho más. El precio es muy variable, dependiendo del tipo de programa.

Un poco a ojo podríamos decirte que los precios más altos oscilan entre 10.000 y 15.000 ptas.

3. Si se trata de software comercial, es un asunto difícil, porque normalmente va protegido. De todas maneras, existen en el mercado programas de utilidad que te lo permitirán, pero no se puede asegurar, a priori, que sea posible con todos.

Soy componente de un piso de estudiantes en Zaragoza. Somos 4 compañeros. Estudiamos Ingeniería Informática.

Nos ha agradado mucho el encontrar en el mostrador de una librería vuestra revista.

Estamos decididos a colaborar con la revista en cuanto podamos.

(Error en pág. 29, en el cuadro bit 14 *8192*).

¿Es verdad que se pueden acoplar al **Amstrad** unidades de disco 5¼?

¿Esa interface sólo vale para la segunda unidad?

¿Necesita algún controlador o puede acoplarse directamente (CPC-464, Interface, Unidad 5¼)?

¿La utilización de la unidad 5¼ sería igual *(en cuanto a SOFT)* que la de 3''?

¿Qué unidades conocidas se podrían conectar?

Gracias.

Rafa Urmeneta (Navarra)

*Sí, se le pueden acoplar unidades de 5¼ al **Amstrad.** En Inglaterra y Alemania existen los aparatos.*

2. Depende del diseño o de la Interface.

3. Normalmente, el disco debiera venir con su propio controlador e interface incluido en el precio, listo para funcionar sin más que enchufarlo.

*4. Lo más lógico es que la unidad de disco de 5¼ funcione bajo CP/M standard, y tal vez también, bajo AMSDOS. En cuanto a soft, esta unidad permitiría acceder a una biblioteca de programas muchísimo mayor que la que **Amstrad** ofrece actualmente, porque existen más de 8.000 programas CP/M, pero en 5¼.*

5. No entendemos muy bien a lo que te refieres con unidades «conocidas». Si te refieres a marcas, ninguna, porque estos productos son nuevos, de muy reciente aparición y algunos desarrollados por particulares. Parece que su proceso de comercialización, sobre todo en nuestro país, va a llevar su tiempo.

De nada.

Soy propietario de un **Amstrad** CPC-464, el cual adquirí por su gran versatilidad y capacidad, que tienen los miniordenadores, para procesar textos. El único inconveniente que le vi era el teclado que, al ser en versión inglesa, carecía de algunos caracteres españoles: ¡, ¿, Ñ, ñ, á, é, í, ó, ú, ü (¡y ¿están definidos en el código ASCII: 175 y 174; pero no tienen tecla asignada). Este detalle no impidió que comprase el ordenador ya que en él es fácil redefinir el teclado y el código de caracteres ASCII.

Una vez redefinidas las teclas que precisé para incluir los caracteres españoles, anteriormente mencionados, intenté trabajar con el procesador de textos que se incluía en el paquete de software de regalo; pero al cargarlo en el ordenador, me restauraba los códigos primitivos deshaciendo el trabajo previo.

Llegado a este punto, mis preguntas son: ¿Hay en el mercado algún procesador de textos que incluya en su programación los caracteres arriba mencionados? ¿Cuál/es? Teniendo en cuenta que las impresoras disponibles trabajarán en versión inglesa, ¿qué tipo de impresoro tendría que comprar para que fuera compatible con el procesador?

Felicitándoles por este nacimiento, y sobre todo por la sección Serie Oro (¡chapó!), se despide de ustedes.

Antonio Alfredo Rivas (Oviedo)

La mayoría de los procesadores de texto de mercado te permitirán definir sin problemas los caracteres especiales de nuestro idioma. Lo que ocurre es que debes preparar a tu impresora para que los reconozca, normalmente mediante lo que se conoce como «secuencias de escape». El manual de tu impresora explicará cómo realizar esto.

Los dos programas de proceso de texto que revisamos en Banco de Pruebas, en números anteriores de la revista, están preparados para funcionar con impresoras compatibles Epson. También pueden funcionar con otras, casi todos en realidad, pero tendrás que «instalar» la impresora de que se trate comunicándoselo al programa mediante un programa especial normalmente llamado «set up», en el que vas contestando a una serie de preguntas que el programa te hace.

AMSTRAD SOFT

P.V.P. (CASSETTE) 1.900: / (DISCO) 2.900:

P.V.P. (CASSETTE) 1.900: / (DISCO) 2.900:

P.V.P. (CASSETTE) 2.500: / (DISCO) 3.300:

P.V.P. (CASSETTE) 2.100: / (DISCO) 3.100:

P.V.P. (CASSETTE) 2.100: / (DISCO) 3.100:

P.V.P. (CASSETTE) 2.300: / (DISCO) 3.300:

P.V.P. (CASSETTE) 2.100: / (DISCO) 3.100:

P.V.P. (CASSETTE) 1.900: / (DISCO) 2.900:

P.V.P. (CASSETTE) 1.900: / (DISCO) 2.900:

P.V.P. (CASSETTE) 1.900: / (DISCO) 2.900:

dissery: TF

ACE DISTRIBUCION
Actividades Comerciales Electrónicas, S.A.
Tarragona, 112 Tel. 325 15 12* Telex 93133 ACEE E 08015 Barcelona

YA DISPONIBLE EN El Corte Inglés ... Y EN TODAS LAS TIENDAS ESPECIALIZADAS

Mercado común

Con el objeto de fomentar las relaciones entre los usuarios de AMSTRAD, *MERCADO COMUN* te ofrece sus páginas para publicar los pequeños anuncios que relacionados con el ordenador y su mundo se ajusten al formato indicado a continuación.

En *MERCADO COMUN* tienen cabida, anuncios de ventas, compras, clubs de usuarios de AMSTRAD, programadores, y en general cualquier clase de anuncio que pueda servir de utilidad a nuestros lectores.

Envíanos tu anuncio mecanografiado a: **HOBBY PRESS, S.A.**
AMSTRAD SEMANAL.
Apartado de correos 54.062
28080 MADRID
¡ABSTENERSE PIRATAS!

Desearía **contactar** con usuarios del CPC-6128 en Córdoba y provincia para intercambio de ideas. Interesados llamar al Tel. (957) 48 10 40 en horas de oficina y preguntar por Pedro.

Deseo **contactar** con usuarios del **Amstrad** CPC-464 para intercambio de programas, de ideas, etc. Enrique. Apdo. de Correos 522. 12000 Castellón. Tel. 20 76 97. Llamar entre las 12,00 y las 16,00 h.

Intercambio todo tipo de programas de **Amstrad** en cinta. Estoy interesado en especial en programas de gestión y utilidades (también los compraría). Dispongo de más de 50 programas. Contesto a todos. Escribe a: Luis Lama Carmona. Avda. de Nazaret, 3 - 2.° B. 28009 Madrid.

Desearía **contactar** con usuarios del **Amstrad** CPC-464, 664, 6128 para el intercambio de programas, ideas, trucos... Preferentemenen en Logroño *(La Rioja)*. Eduardo Lahera Martínez. C/ Rca. Argentina, 3 - 2.° drcha. C.P. 26002. Tel. (941) 23 71 10.

Desearía **contactar** con usuarios de **Amstrad** para intercambio de programas e información general. (Tengo muchos programas en disco). José L. Rojo. Tel: (983) 29 41 04. O por carta: José L. Rojo Matarranz. C/ Tórtola, 9 - 1.° C. 47012 Valladolid.

Vendo Amstrad CPC-464 pantalla color e impresora **Amstrad** DMP-1 matricial 80 col. y 480 col. modo gráfico 50 C.P.S. Precio nuevo actual 168.000 ptas. **Vendo** por 140.000 ptas. Tel. (96) 332 32 83. Valencia.

Vendo Spectrum plus, interface 1, microdrive, interface tipo Kempston y más de 100 programas de utilidades y juegos (ultimate, micro-gen, O.C.P., etc.) en cartuchos y cassettes. Todo con instrucciones y manuales. Como nuevo, 39.000 ptas. Juanjo. Tel. (91) 433 28 57.

Deseo **relacionarme** con usuarios de **Amstrad** CPC-6128/664 para intercambiar listados y experiencias. Escribir a Juan A. Blanco García. Avda. Tomás Giménez, 29. Ent. 2. Hospitalet *(Barcelona)*.

Amstrad Ideas

AMSTRAD Semanal comunica a todos sus lectores la apertura de una nueva sección dedicada a recoger las mejores ideas que exploten al máximo las posibilidades del ordenador, materializadas en programas claros y cortos *(máximo 25 líneas)*. Los mejores de entre todos ellos serán publicados con el nombre de su autor en la revista, recibiendo como premio, gratuitamente en su domicilio los cuatro primeros números de nuestra cinta mensuol. Los programas enviados deberán incluir:

— Cinta de cassette con el programa o programas grabados.

— Explicación detallada del funcionamiento y propósito del programa, mecanografiado a 2 espacios o con letra clara.

Es imprescindible indicar en el sobre claramente: **AMSTRAD IDEAS.**

La dirección es:
Hobby Press, S. A.
Lo Granja, s/n.
Polígono Industrial de Alcobendos.
Madrid

Estoy **interesado** en adquirir monitor en color para **Amstrad** CPC-464. Precio a convenir. Dirigirse a: Eulogio Marzo. C/ Lérida, 9 - 1.°-2.°. Sant Vicenç dels Horts. Barcelona. Llamar al Tel. (93) 656 39 78.

Vendo Amstrad 464 monitor color con manuales en castellano, joystick y 10 programas (adquirido el 31-7-85 en el Corte Inglés) todo por sólo 75.000 ptas. David Más Miró. C/ Galicio, 15. Apdo. 601. Las Palmas de G. Canarias.

Desearía **intercambiar-comprar-vender** programas para el **Amstrad** CPC-464. Interesados escribir a: Pedro Gifreu Feixas. C/ Mn. Ramón Avellano, 2. Mata *(Gerona)*. Tel. (972) 57 36 73.

MICRO-1

Duque de Sexto, 50. 28012 Madrid
Tels. (91) 274 53 80-276 96 16

SOFTWARE: ¡¡GRATIS 1 BOLIGRAFO DE ACERO CON RELOJ INCORPORADO!!

	Ptas.		Ptas.
Bounty Bob	2.300	Bruce Lee	2.300
Fighting Warrior	2.100	Yier Kung Fu	2.300
Foupack	3.890	Exploding Fist	2.300
Combat Lynx	2.100	Southern Belle	2.300
Dummy Run	2.100	Dragontorc	2.300
Raid	2.300	Rocky	2.100
Match Day	2.300	World Series Baseball	2.100
Map Game	2.750	Dambusters	2.300
Hypersports	2.300	Ajedrez Tridimensional	1.975

IMPRESORAS: ¡¡20% DE DESCUENTO SOBRE P.V.P.!!

Lápiz óptico DK'Tronics	4.850	Diskette 3"	1.050
Tapa metacrilato AMSTRAD	1.975	Cinta C-15 (especial)	85
Toshiba MSX 64 K	39.900	Cassette Especial	5.295

Joystick Quick Shot II
2.495 ptas.

Joystick Quick Shot I
1.995 ptas.

Joystick Quick Shot V
2.995 ptas.

Increíbles precios para tu AMSTRAD 464 y 6128 (verde y color). (Llámanos y te asombrarás)

PC Compatible IBM 256 K
Monitor Fósforo Verde
2 Bocas Diskette 360 K
279.000 ptas.

Sabrewulf + Decathlon + Beach Head + Jet Set Willy
2.500 ptas.

Commodore 64: 42.900 ptas.
Commodore 128: 74.900 ptas.

El pedido te lo enviamos URGENTEMENTE contra-reembolso SIN NINGUN GASTO DE ENVIO, LLAMANDO a los teléfonos: (91) 276 96 16-274 53 80 o escribiendo a MICRO-1. Duque de Sexto, 50. 28012 Madrid.

SACALE EL JUGO A TU ORDENADOR. DISEÑA TUS PROPIAS PANTALLAS Y DIVIERTETE JUGANDO CON...

Este mes:

YOUR COMPUTER

Te ofrece algo realmente sabroso:

MUSICA

Este magnífico programa escrito en código máquina te permitirá manejar el sonido y la música en tu Amstrad desde Basic, mediante un nuevo juego de comandos creados especialmente para ello.

CROSS

Tienes cuatro revólveres para destruir a tus enemigos en el mínimo tiempo posible. Necesitarás toda tu habilidad, rapidez de reflejos y suerte, mucha suerte.

JUMPER

Debes alcanzar la cima del Valle de las Cintas Deslizantes. Tienes que saltar por los huecos de las vallas, que se desplazarán a derecha e izquierda con una rapidez de vértigo.

MAGGOT

Te encuentras en la amable tierra de las setas gigantes. Tu misión es guardarla del ataque y la invasión de una peligrosísima serpiente polimórfica que las ataca sin piedad.

TIMEBOMB

Una organización terrorista de Oriente Medio ha colocado una bomba de tiempo en el laberinto de defensa del Laboratorio de investigación bacteriológica de Lexington.

RSX

Your Computer ha pensado en los usuarios del Amstrad CPC464 y ha creado un nuevo juego de comandos completo para tu ordenador, de forma que el Basic así ampliado no tenga nada que envidiar al de los otros modelos de la serie.

SINTAX, S.A.

Si no lo encontrara en su kiosco, puede solicitarlo directamente a nuestra editorial:
Paseo de la Castellana, 268. Tel.: (91) 733 25 99. 28046 Madrid.